VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Windows Azure a možnosti push notifikací na mobilních platformách

Windows Azure and Features of Push Notifications on Mobile Platforms

2013 Marián Bielik

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Marián Bielik**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Windows Azure a možnosti push notifikací na mobilních platformách
Windows Azure and Features of Push Notifications on Mobile Platforms

Zásady pro vypracování:

Dnes rostoucí obliba mobilních a cloud technologií otvírá řadu implementačních problémů. V této diplomové práci se diplomant zaměří na problematiku push notifikací, které je možno používat na všech nejrozšířenějších mobilních platformách (Androoid, iOS a Windows Phone). Jako server, který bude využíván pro kontrolu notifikací použije cloud řešení Windows Azure.

Jednotlivé body práce jsou:
1. Prostudovat a popsat možnosti push notifikací pro Android, iOS a Windows Phone.
2. Prostudovat a popsat části Windows Azure potřebné pro hostování vyvíjené služeb.
3. Při popisu se zaměřit také na bezpečnost push notifikací a server řešení pomocí Windows Azure.
4. Vyvinout RSS čtečku, který pro různé platformy s možnostmi push notifikací.
5. Provést srovnání jednotlivých platforem a jejich možností push notifikací.
6. Provést experimentální ověření vyvinuté aplikace, včetně výkonnostního testování.

Seznam doporučené odborné literatury:

[1] Neil Mackenzie: Microsoft Windows Azure Development Cookbook, Packt Publishing, 2011
[2] Cloud service for pushing android notifications, http://www.push-notification.org/
[3]- Using Local and Push Notifications, https://developer.apple.com/appstorepush-notifications/index.html
[4] Push Notifications Overview for Windows Phone, http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff402558(v=vs.92).aspx

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jan Martinovič, Ph.D.**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave dňa 1.5.2013 .....

Poďakovanie

Na tomto mieste by som rád poďakoval všetkým, ktorí mi s prácou pomohli, pretože by bez nich tato práca nevznikla.

Abstrakt

Diplomova práca sa zaoberá push notifikáciami pre rôzne platformy a to Windows Phone7, Windows Phone 8, Android, Windows 8, iOS. Porovnáva možnosti notifikácii v rámci platforiem. Implementáciu aplikácii pre tieto platformy a implementáciu do prostredia Windows Azure s využitím možností tohto prostredia. Zaoberá sa bezpečnosťou push notifikácii pre rôzne platformy a výkonnostnými testami notifikácii.

Kľúčové slová

Push notifikácie, Windows Azure, iOS, Android, Windows Phone 7, Windows Phone 8, Android, Windows 8

Abstract

This Diploma Thesis deals with push notifications for various platforms like Windows Phone 7, Windows Phone 8, Android, Windows 8, iOS. Compares the notification options within platforms. Implementation of applications for these platforms and implementation into Windows Azure environment using the possibilities of this environment. It deals with safety of push notifications for different platforms and performance tests of notifications.

Keywords

Push notifications, Windows Azure, iOS, Android, Windows Phone 7, Windows Phone 8, Android, Windows 8

Zoznam použitých symbolov a skratiek

NDK	-	Native Development Kit
JDK	-	Java Development Kit
ADT	-	Android Development Toos
GCM	-	Google Cloud Messaging
C2DM	-	Cloud to Device Messaging
MPNS	-	Microsoft Push Notification Service
WNS	-	Windows Notification Service
URI	-	Uniform Resource Identifier
GUID	-	Globally Unique Identifier
VM	-	Virtual Machine
APNS	-	Apple Push Notification Service
CAS	-	Code Access Security
WCF	-	Windows Communication Foundation
HTTP	-	Hypertext Transfer Protocol
TCP	-	Transmission Control Protocol
IP	-	Internet Protocol
REST	-	Representational State Transfer
JSON	-	JavaScript Object Notation
SOAP	-	Simple Object Access Protocol
SSL	-	Secure Sockets Layer
XML	-	eXtensible Markup Language
IIS	-	Internet Information Services
DNS	-	Domain Name System
NTFS	-	New Technology File System

Obsah

Zoznam tabuliek

Zoznam obrázkov

Zoznam výpisov zdrojového kódu

1 Úvod

Push notifikácie sú čoraz populárnejšie a v tejto diplomovej práci sa budem venovať práve ich porovnaniu a implementácii pre rôzne platformy. Server, ktorý bude posielať notifikácie, bude pre všetky platformy rovnaký a nasadený na Windows Azure. Na tomto serveri budem voliť architektúru, ktorá bude najviac vyhovovať aplikácii na posielanie push notifikácii.

Architektúra push notifikácii je rôzna pre jednotlivé platformy. Z veľkej časti ide o zabezpečenú HTTPS komunikáciu medzi zariadením a príslušným notifikačným serverom alebo samotným koncovým zariadením a notifikačným serverom. Každá platforma ale pristupuje k zabezpečeniu pomocou HTTPS odlišne a vyžaduje rozdielnu implementáciu a autentifikáciu, ktorej sa budem v tejto práci venovať.

Výsledkom práce by mali byť aj výkonnostné testy, ktoré budú využívať naimplementované notifikácie pre rôzne platformy.

1.1 Štruktúra práce

Popis jednotlivých platforiem, ich vývojových nástrojov a popis notifikácii si predstavíme v kapitole 3. Sekcia 3.6 porovnáva predstavené notifikácie. Samotnou implementáciou sa budeme zaoberať v kapitole 4. Sekcia 4.5 porovnáva notifikácie z pohľadu implementácie. Kapitola 5 predstavuje Windows Azure. Bezpečnosťou notifikácii sa zaoberá kapitola 6. Výkonnostnými testami naimplementovaných notifikácii sa zaoberá kapitola 7.

2 Popis notifikácii

Väčšina dnešných mobilných aplikácii závislých na nepravidelnom získavaní informácií z webového serveru, ako napríklad Facebook, Gmail, používa takzvané push notifikácie. Prvý krát boli vytvorené firmou Apple, ktorá ich nasadila spolu s operačným systémom iOS 3.0 (mobilný operačný systém) a to 17.6.2009, ako sa uvádza v [1].

Je to služba serveru danej platformy pre notifikácie určeného (APNS, GCM, WNS,…), ktorá autorom aplikácii dovoľuje na základe podnetu od iného serveru poslať krátku správu, ktorá môže dať napríklad vedieť aplikácií, aby sa pozrela na vlastný server a stiahla si z neho aktuálne informácie, alebo informovať užívateľa, formou krátkej správy, tónu, obrázku o novej udalosti.

Tento prístup ma z princípu určité výhody. Najmä nevyžaduje, aby samotná aplikácia periodicky kontrolovala vlastný server, či na ňom nie sú nové dáta, čo pomerne výrazne šetrí batériu. Technológia je dostupná zdarma pre všetky platformy a celkom ľahko implementovateľná. Samotná aplikácia má otvorené IP spojenie so svojím notifikačným serverom, od ktorého prijíma notifikácie.

Bohužiaľ nie je stopercentne spoľahlivá. Závisí totiž na trvale udržovanom internetovom pripojení. A práve to môže občas vypadnúť, čo vo výsledku znamená, že všetky aplikácie, ktoré používajú push notifikácie prestanú dostávať správy o novom obsahu. Po znovu pripojení na internet sa toto spojenie môže vytvoriť opäť, ale ak sa aplikácia počas tohto obdobia odinštaluje, notifikáciu už nedostane a samotný notifikačný server nijako neinformuje server tretej strany, ktorý žiadal o odoslanie push notifikácie, či bola jeho správa doručená. Túto funkcionalitu si už musí doprogramovať autor aplikácie podľa jej potrieb.

Notifikácie podporujú takmer všetky mobilné operačne systémy ako je iOS, Windows Phone, Android, Bada,... aj niektoré nové desktopové operačné systémy ako Mac OS X Lion, Windows 8 a jeho nové rozhranie Modern UI, ako sa uvádza v [2].

Push notifikácie majú svoje obmedzenia, do veľkosti správy, zabezpečenia a určite by nemali obsahovať citlivé dáta užívateľov.

3 Použité platformy

V nasledujúcich kapitolách sa budeme zaoberať popisom použitých plaforiem.

3.1 Android

Android je rozsiahla open source platforma najmä pre mobilné zariadenia, vyvíjaná spoločnosťou Google. Operačný systém je založený na jadre z Linuxu, middleware. Prvý mobilný telefón s Androidom, pôvodne vo verzii 1.0, sa začal predávať na trhu koncom roka 2008. Súčasná verzia je 4.3, ako je uvedené v [3].

Aplikácie nekomunikujú priamo s jadrom, ale cez jednotné Android API. Takto aplikácie pristupujú k funkciám telefónu ako je displej, senzorom, GPS, kompasu,... O samotný beh aplikácii sa stará Dalvik Virtual Machine, čo je virtual machine podobná Java Virtual Machine.

Obrázok 1: Android logo

Programovací jazyk pre aplikácie je Java, alebo je možné programovať aj priamo v natívnom kóde v C/C++ označovaného ako, Native Development Kit (NDK).

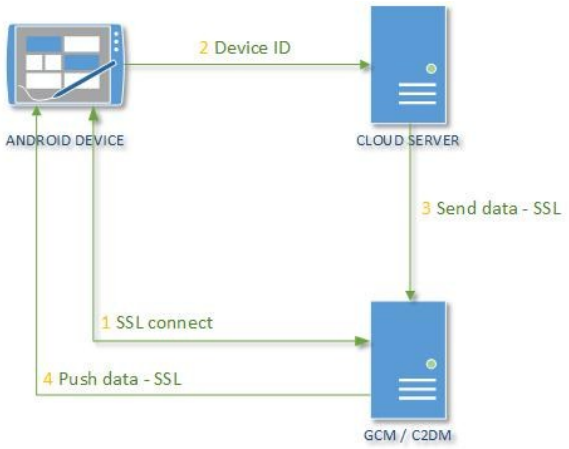
Oficiálny nástroj pre vývoj aplikácii je Eclipse IDE, ktorý je zdarma. K nemu je potrebný Java Development Kit (JDK) a Java Software Development Kit (SDK) pre Android. Pre vývoj je potrebný aj Android Development Toos (ADT), čo je plugin pre Eclipse, ktorý pridá možnosti týkajúce sa Androidu. Stiahnutie konkrétnej pratformy umožňuje SDK Manager, ktorý zobrazuje dostupné platformy. Pre testovanie notifikácii je v tejto diplomovej práci stiahnutá do Eclipse verzia 2.3, 3.0 a 4.0 a potrebné Google Cloud Messaging for Android Library.

Aplikácia, ktorá používa push notifikácie musí byť zaregistrovaná na Google APIs, kde sa vytvorí projekt pre našu aplikáciu. Samotná podpora push notifikácii sa sprístupní na záložke services po zaškrtnutí Google Cloud Messaging (GCM).

Ďalšia služba potrebná pre push notifikácie je Google Account. Každý užívateľ aplikácie, ktorá podporuje push notifikácie, musí mať prihlásený Google užívateľský účet na svojom zariadení. Bez tohto účtu sa ani neprihlási so Google Marketu, aby si stiahol aplikáciu, ale je možné aplikáciu nainštalovať aj bez Marketu, pokiaľ je v zariadení povolená inštalácia aplikácii z neznámych zdrojov.

3.1.1 Architektúra notifikácii a druhy notifikácii

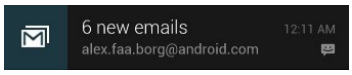
Android zariadenie pomocou svojho Google Account a Project Number pre market (z Google APIs) pošle request na Google GCM, alebo starší Cloud to Device Messaging (C2DM) server, z ktorého získa Device ID potrebné pre notifikácie.



Obrázok 2: Android architektúra

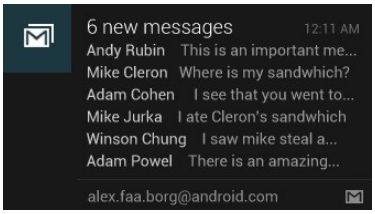
Následne je ID poslané na server, ktorý nám bude generovať notifikácie pre GCM server.
Tuto notifikáciu spolu s Device ID a API key (z Google APIs) pošle náš server na google GCM. Na toto API key sa v Google APIs dá nastaviť list IP adries, z ktorých google server umožní posielanie notifikácii pre danú aplikáciu a tým aj ich filtrovanie.
GCM server odpovie HTTP statusom 200 ak sú všetky údaje v poriadku a odošle túto notifikáciu na zariadenie s rovnakým Device ID.

Notifikácie sa zobrazujú v takzvanom Normal view, ako je uvedené v [4]. O samotné vyplnenie notifikácie dátami sa stará aplikácia po prijatí notifikačnej správy a aj o jej samotné vyvolanie.



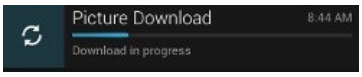
Obrázok 3: Android Normal view notifikácia

Ďalší typ je notifikácie je Big view, ktorý je dostupný od verzie 4.1 a zobrazuje sa až pri detailnom otvorení notifikácie, inak zostáva zobrazený ako Normal view. Rovnako sa o vyplnenie a zobrazenie stará aplikácia, respektíve metóda volaná pri prijímaní notifikácie.



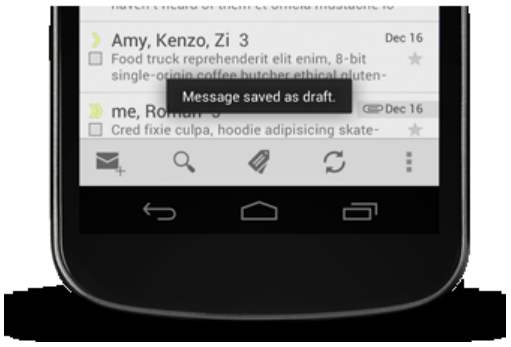
Obrázok 4: Android Big view notifikácia

Pomocou notifikácii sa dá zobraziť aj progres bar. Využitie je možné napríklad pri sťahovaní dát. Po skončení sťahovania je možne tuto notifikáciu zmeniť. Všetky typy sa dajú aktualizovať, odoberať.



Obrázok 5: Android Progres bar notifikácia

Ďalším typom sú Toast notifikácie, ktoré sú určené ako jednoduchá spätná väzba o nejakej operácii v podobe malého vyskakovacieho okna. Automaticky sa zavrú po predom určenom časovom limite.



Obrázok 6: Android Toast notifikácia

3.2 Windows Phone 7 a Windows Phone 8

Windows Phone je mobilný operačný systém Microsoftu. Jedná sa o nástupcu systému Windows Mobile, s kterým ale nieje spätne kompatibilný. Windows Phone bol prvý krát vydaný 21.10.2010, ako je uvedené v [5]. Bola to verzia Windows Phone 7. V súčasnosti je najnovšia verzia Windows Phone 8, ktorá je založená na jadre z rady Windows NT na rozdiel od predchádzajúceho Windows Phone 7, respektíve 7.8, ktorý je založený na Windows CE. To znamená aj to, že nie sú spätne kompatibilné. Užívateľské rozhranie zložené z dlaždíc a je pomenované Modern UI.

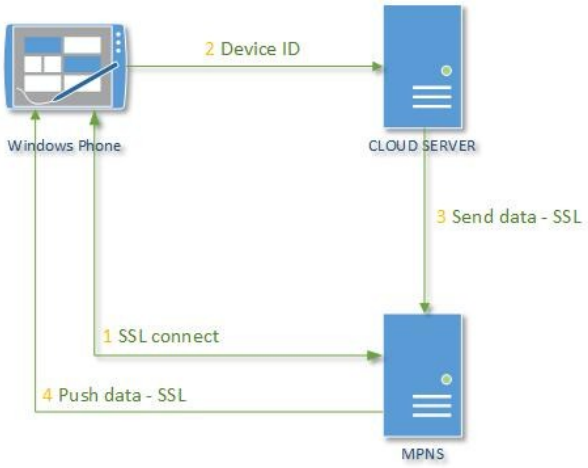
Obrázok 7: Windows Phone logo

Programovací jazyk primárne C# alebo VB.NET a C++, ktoré je obmedzené len pre hry. Vo Windows Phone 7 neexistuje varianta XAML a C++ alebo HTML5 a JavaScript ako na Windows 8. To neplatí pre Windows Phone 8.

Oficiálny nástroj pre vývoj je Visual Studio do ktorého treba doinštalovať Software SDK. Pre Windows Phone 7 a 7.5 je to SDK 7.1, pre Windows Phone 8 je to SDK 8. Ja som začal s Windows Phone 7, pretože Windows Phone 8 bol vydaný až ku koncu tvorby mojej diplomovej práce. Aplikácie pre Windows Phone 7 sú funkčné aj na Windows Phone 8.

K samotnému vývoju je potrebný developerský účet Microsoftu, ktorý stojí 99$, ale pre študentov vlastniacich ISIC, je pre nekomerčné účely zdarma na developerskom portále Microsoftu. Stačí mat Live účet, pretože tento developerský účet sa viaže na Live účet. Žiadne ďalšie služby nie sú potrebné pre začatie vývoja aplikácii pre nekomerčné účely.

3.2.1 Architektúra notifikácii a druhy notifikácii



Obrázok 8: Windows Phone architektúra

Windows Phone je starší ako Windows 8, preto je aj architektúra notifikácii staršia a aj zložitejšia na implementovanie. Pre získanie URI, ktoré slúži na identifikáciu zariadenia, z Microsoft Push Notification Service (MPNS), pošle aplikácia svoje Device ID vo forme GUID a následne získa URI. Získané URI zašle aplikácia na server. Ak všetko prebehne v poriadku, po odoslaní

notifikácie server vráti kód 200, čo znamená, že všetko prebehlo v poriadku. O doručenie sa samozrejme stará samotný MPNS server.

Druhy notifikácii sú **Toast:**

Sú to krátke, textové správy, ktoré sa zobrazujú v hornej časti obrazovky na krátku, približne 10 sekúnd, aby upozornili užívateľa zvukom aj obrazom o novo vzniknutej udalosti. Majú pevnú veľkosť, nie je možné na nich zobrazovať niečo iné ako text. Text sa zobrazuje spolu s ikonou aplikácie. Kliknutím na notifikáciu sa spustí príslušná aplikácia, ktorá dostala notifikáciu. Text zobrazený na notifikácii sa skladá z dvoch časti a to nadpisu a samotného textu správy, ktoré sa zobrazia za sebou. Nadpis môže mať maximálne 40 znakov a samotný text 47, zvyšné znaky sa nezobrazia.

Obrázok 9: Windows Phone Toast notifikácia

Tile:

Tile notifikácia je dlaždica na štartovacej obrazovke Windows Phone. Všetky aplikácia majú aspoň jednu Tile, známu ako defaultnu, ktorá sa zobrazuje na štartovacej obrazovke, pokiaľ si ju tam užívateľ pridá. Aplikácie možu mať ďalšie druhotné Tiles. Napríklad aplikácia počasie môže mať tiles priradené k určitému mestu a zobrazí sa podľa toho, aké mesto si užívateľ zvolí.

Podporujú rôzne veľkosti, zvolené užívateľom. Tu je rozdiel v novej verzii Windows Phone 8, kde je možné zvoliť jednu z troch veľkostí a to malej, ktorá má 159 pixelov, strednej, ktorá ma 336 pixelov a veľkej, ktorá má 691 pixelov. U staršej verzie Windows Phone 7, malá veľkosť nie je dostupná. Na notifikácii sa okrem textu a pohyblivého obrázku môže zobrazovať aj počet napríklad neprečítaných nových udalostí.

Obrázok 10: Windows Phone Tile notifikácia

Windows Phone 8 podporuje tri šablóny: flip, iconic a cycle. Programátorsky sa zvolená šablóna už nedá zmeniť. Flip šablóna je ako jediná podporovaná staršou verziou Windows Phone. Obrázok zobrazovaný na dlaždici musí byť menší ako 80KB, sťahovanie tohto obrázku nesmie presiahnuť 60 sekúnd.

Raw:

Raw notifikácie sú podobné tile a toast notifikáciám až na to, že sa nemajú svoj zobrazovací formát, alebo šablónu ako majú vyzerať. Aplikácia musí byť spustená, aby mohla prijať tento druh notifikácie. Ak nie je spustená, tak je ignorovaná samotným systémom. Aplikácia musí byť prispôsobená na prijímanie tohto druhu notifikácii za behu. Vo Windows 8 majú raw notifikácie iný význam a fungujú iným spôsobom.

3.3 iOS

Mobilný operačný systém iOS vytvorený spoločnosťou Apple. Pôvodne bol určený len pre mobilné telefóny iPhone, neskôr sa však začal používať aj na ďalších mobilných zariadeniach tejto firmy, ako je iPod Touch, iPad a najnovšie aj Apple TV. Prvá verzia bola uvedená na trh 29.6.2007 pre prvý iPhone. Posledná verzia je iOS 6, ktorá bolá uvedená 11.6.2012, ako je uvedené v [6]. Meno iOS dostal od štvrtej generácie tohto systému. Do tej doby bol oficiálne nazvaný iPhone OS.

iOS je odľahčená verzia operačného systému Mac OS X, používaného v počítačoch spoločnosti Apple. Jedná sa teda o systém UNIXového typu. Keďže je určený pre mobilné zariadenia, neobsahuje všetku funkcionalitu OS X.

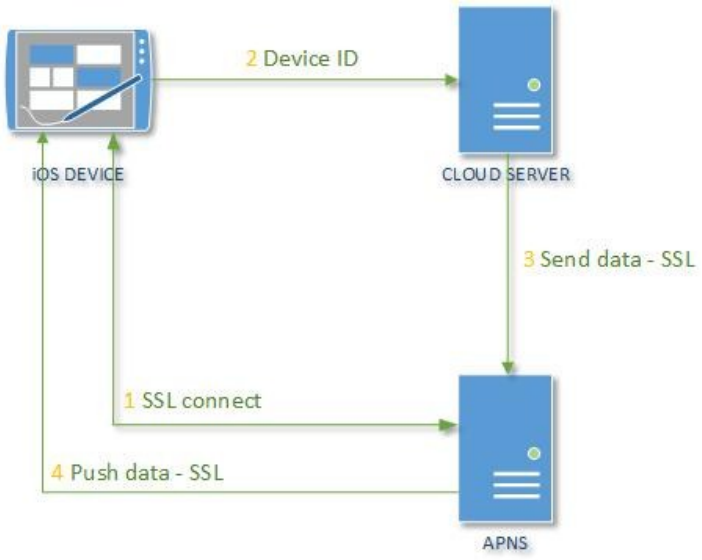
Obrázok 11: iOS logo

Aplikácie je možné programovať v jazyku C alebo Objective-C. Vývojovým nástrojom pre tento systém je XCode, ktorý ale potrebuje pre svoj beh operačný systém Mac OS X. Tento systém ale obsahujú len zariadenia spoločnosti Apple. Keďže ale nevlastním žiadne zariadenie s týmto systémom, pokúsili sme sa ho nainštalovať pomocou programu VMware ako virtuálny na svoj osobný počítač s AMD procesorom. Po pár dňoch neúspešných pokusoch nainštalovať rôzne verzie tohto operačného systému sme sa rozhodoli ho nainštalovať do VMware na notebook s procesorom poslednej verzie spoločnosti Intel a to Sandy Bridge, pretože aj produkty spoločnosti Apple používajú tento systém. Už druhá verzia, ktorú sme sa pokúšali nainštalovať, sa nainštalovala úspešne.

Pre vývoj na mobilné zariadenia je potrebný XCode, ktorý je potrebné doinštalovať z Apple Storu. Ten ale vyžadoval verziu Mac OS X, ktorú sme nemali a našu verziu sa samozrejme nepodarilo aktualizovať, keďže sme nemali zariadenie Apple. Preto sme museli nájsť konkrétnu verziu Mac OS X pre XCode. Po úspešnom nájdení a nainštalovaní tohto systému sme nainštalovali aj XCode z Apple Storu a vytvoril a spustil jednoduchú aplikáciu na tomto notebooku s procesorom od firmy Intel.

Pre vývoj je potrebný len samotný XCode, pomocou ktorého sa dajú vytvoriť jednoduché aplikácie, ktoré nie sú umiestnené na Apple Store. Na to treba mať zakúpený developerský účet, ktorý stojí 99$ na rok a nie je jeho študentská verzia. Kôli potrebe certifikátu na notifikácie musíme vlastniť platený developerský účet. Tento ale nemáme, preto sme notifikácie z Windows Azure nemali kam posielať a na tejto platforme bude táto práca popisovať len serverovú aplikáciu. Clientska len do tej miery, kým si nevyžadovala aplikácia overený certifikát od spoločnosti Apple.

3.3.1 Architektúra notifikácii a druhy notifikácii



Obrázok 12: iOS architektúra

iOS aplikácia požiada o takzvaný token, aplikácia musí byť nakonfigurovaná, aby podporovala push notifikácie na developerskom portále Apple, Apple Push Notification Service (APNS) server, z ktorého získa token potrebný pre notifikácie. Následne je token poslaný na server, ktorý nám bude generovať notifikácie pre APNS server. Na rozdiel od Androidu, pre poslanie notifikácie je potrebný certifikát vytvorený na Apple portále a ID aplikácie. Pomocou tohto certifikátu a unikátnemu tokenu pošle server notifikáciu na APNS server, ktorý sa postará o jej doručenie.

Druhy notifikácii sú **Alert:**

Táto správa pozostáva z názvu aplikácie, krátkej správy a prípadne dvoch tlačítok. Tlačidlo na pravej strane je nazvané tlačidlo akcie a má názov Zobraziť. Aplikácia môže zmeniť názov tohto tlačidla a prípadne tak previesť preklad tohto názvu do konkrétneho jazyka zvoleného v aplikácii. Pokiaľ užívateľ zvolí toto tlačidlo, spustí sa daná aplikácia, stiahnu sa nové dáta. Naopak ľavé tlačidlo zavrie notifikáciu. Tento alert sa môže zobrazovať aj bez tlačidiel a po určitom čase vyprší. Môže zobrazovať aj len jedno tlačidlo, s funkcionalitou vyžadujúcou potvrdenie o tom, že sme si notifikáciu prečítali a po zvolení tohto tlačidla sa aplikácia zavrie. Táto notifikácia môže upozorňovať aj zvukom a upozorňuje užívateľa len ak je spustená.

Obrázok 13: iOS Alert notifikácia

Badge:

Tento druh notifikácie zobrazuje číslo v hornom pravom rohu, napríklad počet neprečítaných emailov, udalostí v kalendári a iné. Tento druh notifikácie je jediný, ktorý funguje pre nespustené aplikácie. A zobrazuje sa teda len keď aplikácia nie je spustená.

Obrázok 14: iOS Badge notifikácia

3.4 Windows 8

Je nástupcom Windows 7 od 26.10.2012. Windows 8 je operačný systém, ktorý je určený pre použitie v desktopoch aj tabletoch. Bola vydaná verzia pre architektúru x86-64 aj mobilný ARM. Používa užívateľské rozhranie Modern User Interface, ktoré je známe zo systému Windows Phone. V systémových aplikáciách sa rozšírilo vysúvacie menu Ribbon, ktoré nahrádza klasické textové menu.

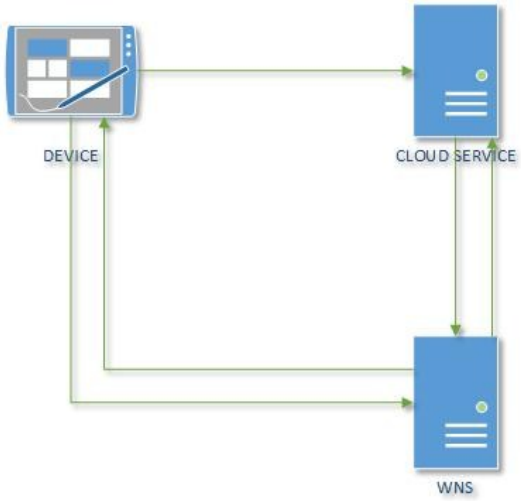
Obrázok 15: Windows 8 logo

Vývoj aplikácii pre Windows 8 Modern User Interface, takzvaných Windows Store apps, je potrebné mať nainštalované Visual Studio 2012 a SDK 8. To nie je možné nainštalovať na operačný systém Windows 7, takže pre toto nove SDK je potrebné mať nainštalovaný aj samotný systém Windows 8.

K samotnému vývoju je potrebný developersky účet Microsoftu, ktorý stoji 99$, ale pre študentov vlastniacich ISIC, je pre nekomerčné účely zdarma. Stačí mat Live účet, pretože tento developersky účet sa viaže na tento účet. Pri samotnom vývoji som potreboval vytvoriť aplikáciu na Windows Store, aby som získal údaje potrebné pre autentifikáciu notifikácii, ale už pri prvom kroku chcel Microsoft aktualizovať developersky účet zo študentského na verziu umožňujúcu komerčné použitie, to znamená platenú. Študentský je možné využívať len v rámci svojho developerkeho prostredia a systému, nie Windows Storu. Preto som sa k samotnému posielaniu notifikácii z Azure serveru nedopracoval. Samozrejme sme ale aj túto časť naprogramovali, ale notifikácie fungovali len v rámci nášho systému, to znamená, že si ich aplikácia posielala sama.

3.4.1 Architektúra notifikácii a druhy notifikácii

Princíp je podobný ako pre Windows Phone. Zariadenie najskôr získa URI z Windows Notification Service (WNS). Následné je potrebne klient ID (SID) a klient secret, ktoré sa získa zaregistrovaním aplikácie na Windows Store. Tieto získané údaje sú odoslane na server, ktorý ich použije, aby od WNS získal prístupový kľuč, ktorý bude používať na odosielanie notifikácii. Nakoniec pošle samotnú notifikáciu na WNS s použitím URI a prístupového kľúča. O doručenie notifikácie na klienta sa stará samotný WNS server.



Obrázok 16: Windows 8 logo

Druhy notifikácii sú **Badge:**

Zobrazuje predovšetkým číslo na pravej spodnej strane dlaždice z intervalu 1 až 99, rovnako pre úzku aj širokú dlaždicu. 0 znamená žiadna a akékoľvek číslo väčšie ako 99 sa zobrazí ako 99+. Badge notifikácia ma informovať o pocte napríklad: nových emailov, nových správ, alebo dostupných aktualizácii.

Obrázok 17: Windows 8 Badge notifikácia

Okrem čísla môžeme použiť aj nejakú z dostupných takzvaných "Glyphs":

Príkladom pre použitie takejto notifikácie je napríklad "paused" pri pozastavení prehrávania hudby.

Tile:

Sú to dlaždice na štartovacej obrazovke Windows 8. Môžu byt štvorcové, obdĺžnikové, rôznych veľkosti. Tile notifikácie sú pravé tie, ktoré dávajú dlaždiciam ich charakteristicky vzhľad. Je možné na nich zobrazovať textové správy rôznej veľkosti písma, obrázky, taktiež rôznych veľkosti, kombinovať text a obrázky. Obrázky sa môžu na dlaždici dynamicky meniť, takzvane

live tiles, v rôznych intervaloch. Môžeme si na nich rýchlo precitať predmet emailu, statusy na sociálnych sieťach, vidieť nove pridane fotografie…

Obrázok 18: Windows 8 Tile notifikácia

Obrázky musia mat menšiu alebo maximálnu veľkosť 1024x1024 pixelov a maximálne 200KB. Podporovane typy sú png, jpg, jpeg a gif. Bitová hĺbka a farby Nisu nijako obmedzovane. Štartovacia obrazovka podporuje obrázky v tíchlo veľkostiach:

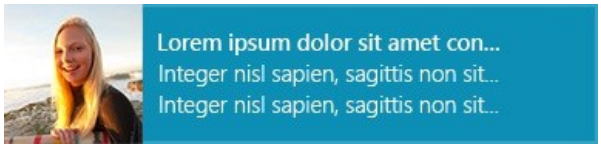
Škála	Široká	Štvorcová
80%	248 x 120	120 x 120
100%	310 x 150	150 x 150
140%	434 x 210	210 x 210
180%	558 x 270	270 x 270

Tabuľka 1: Windows 8 a veľkosť obrázkov

Pokiaľ má užívateľ zvolenú užšiu variantu dláždite a je dostupný len obrázok pre širšiu variantu, zobrazí sa obrázok ktorý je nastavený ako štandardný pre užšiu variantu. Uvedené obmedzenia platia rovnako aj pre toast notifikácie.

Toast:

Toast notifikácie sa zobrazujú v hornej pravej strane obrazovky na krátku dobu, aby upozornili užívateľa obrazom aj zvukom o novej udalosti. Majú pevnú veľkosť a farbu pozadia (podľa nastavenia systému), ale je možné na nich zobraziť obrázok a text. Tento druh notifikácie sa zobrazuje ak keď nemáme pravé zobrazenú štartovaciu obrazovku systému, ale klasicky desktop Windows.



Obrázok 19: Windows 8 Toast notifikácia

Dĺžka zobrazenia na obrazovke závisí od nastavenia, ktoré môže byť buď krátke, alebo dlhé. Krátke sa zobrazujú po dobu siedmych sekúnd a prehráva systémom definovaný zvuk pre tento druh notifikácii. Dlhé sa zobrazujú 25 sekúnd a zvuk môže byt prehrávaný cyklicky. Tento druh môže byt typicky pre oznámenia ktoré čakajú na užívateľove zareagovanie ako je napríklad hovor VOIP. Na rozdiel od toho sú krátke zobrazovane typicky v situáciách ako napríklad pripojenie novej osoby na do komunikačného klienta. V systéme Windows sa takýmto spôsobom zobrazujú novo pridane zariadenia.

Raw:

Tento druh notifikácii nemajú UI zobrazenie pre užívateľa, ani ho nemajú nijakým spôsobom upozorňovať. Používajú sa ako úlohy na pozadí aplikácie, kým je tato aplikácia vypnutá. Typicky sa to dá využiť ako stiahnutie napríklad príloh emailu na pozadí po príchode napríklad Tile notifikácie. Užívateľ neskôr po spustení aplikácie, ktorá už nemusí mať prístup do internetu zobrazí novy email aj s prílohami, ktoré by inak museli byt stiahnuté až po spustení aplikácie, ktorá už nemusí mať prístup do internet. Alebo užívateľ jednoducho nemusí čakať na stiahnutie prílohy, pretože raw notifikácia ich stiahla miesto neho.

Lock screen notifications:

Tieto notifikácie je treba pri vývoji povoliť v aplikácii. Notifikácia bude zobrazená užívateľovi aj pri zamknutej obrazovke Windows 8, kedy sa tile a badge notifikácie nezobrazujú. Toto nastavanie sa musí zapnúť aj v nastaveniach Windows 8 v sekcii personalization. Obrázok ktorý sa zobrazí musí byť maximálnej veľkosti 24x24 pixelov, taktiež zvolený v tejto sekcii. Po prijatí notifikácie sa zobrazí badge notifikácia so zvoleným obrázkom a tile text.

3.5 Windows Azure

Windows Azure je spoločný server pre generovanie notifikácii pre všetky platformy. Nasledujúce podkapitoly sa venujú popisu tejto platformy

3.5.1 Popis Windows Azure

Platforma Windows Azure je internetová platforma pre služby cloud computing hrsťované v dátových centrách spoločnosti Microsoft. Platforma Windows Azure poskytuje celú škálu funkcií pre vytváranie aplikácií, ktorých použitie pokrýva spektrum od spotrebiteľských webov až po

nasadenie v podnikoch a zahŕňa operačný systém pre služby prostredia cloud a sadu služieb pre vývojárov a IT profesionálov. Hlavnými súčasťami platformy Windows Azure sú produkty Windows Azure, Microsoft SQL Azure a AppFabric ako je uvedené v [7].

Podporuje celú sadu jazykov a môžeme ju integrovať s existujúcimi systémami hosťovanými interne vo firme. Pri vývoji aplikácií a služieb na platforme Windows Azure môžu vývojári vychádzať z existujúcich znalostí produktu Microsoft Visual Studio. Windows Azure podporuje najrozšírenejšie štandardy a protokoly ako SOAP, REST, XML alebo PHP.

Obrázok 20: Windows Azure logo

Výhody platformy Windows Azure

- Nižšie nároky na správu IT a na miesto v budovách.
- Rýchlejšia možnosť reagovať na obchodné potreby a požiadavky zákazníkov.
- Škálovateľnosť IT prostriedkov.
- Výpočetné prostriedky je možné zaktívniť na dobu, počas ktorej budú potrebné.
- Žiadna správa hardvéru.
- Využitie existujúcich znalosti pri vývoji

Funkcie platformy Windows Azure:

Výpočty

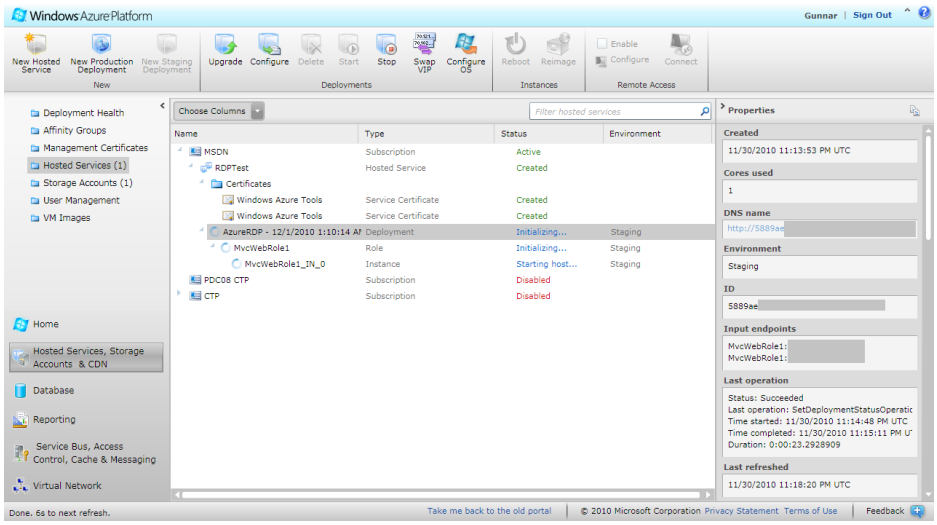
- Možnosť spúšťať webové aplikácie používajúce technológie Microsoft ASP.NET, Java, PHP v cloude.
- Prostredie pre hosťovanie služieb, ktorého súčasťou sú IIS 7 a Microsoft .NET Framework.
- Zabezpečenie podporené flexibilnými zásadami Code Access Security (CAS).
- Webový portál, ktorý uľahčuje a urýchľuje nasadenie, škálovanie a aktualizáciu vašich služieb.
- Podpora technológie FastCGI umožňuje zákazníkom nasadiť a prevádzkovať webové aplikácie, ktoré boli napísané v programovacích jazykoch od iných dodávateľov ako spoločnosť Microsoft (napr. PHP).
- Možnosť vystaviť v cloude webové služby pre externé aj interné aplikácie rozhraním Windows Communication Foundation (WCF).
- Možnosť volať v cloude v aplikáciách aj metódy z natívnych knižníc DLL.

Jednoduché ukladanie dát

- Tabuľky, fronty aj objekty typu blob sú hosťované v prostredí cloud, teda blízko od miesta, kde prebiehajú vaše výpočty.
- Bezpečnosť dát je zaistená vďaka overovaniu prístupu a replikácii.
- Jednoduché rozhrania REST poskytujú nenáročný prístup k dátam a sú k dispozícii vzdialene aj zvnútra dátového centra.

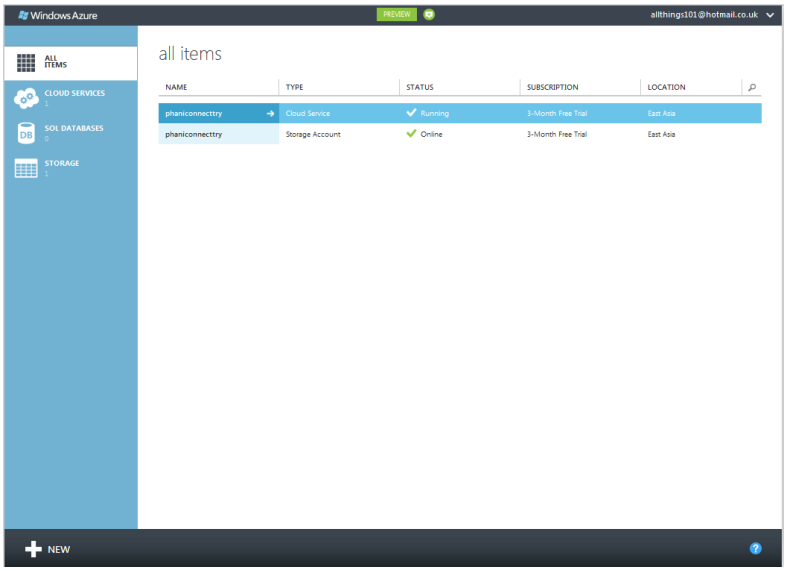
3.5.2 Windows Azure Portál

Práca na portáli, ktorý je nasadený od roku 2010 a spúšťa sa v prehliadači pripomína svojím rozložením kancelársky balík Microsoft Office 2010, pretože oba obsahujú menu, ktoré sa nazýva Ribbon.



Obrázok 21: Windows Azure starý portál

Koncom roka 2012 Microsoft nasadil nový portál, ktorý bol výrazne zmenený. Ribbon Microsoft zrušil a mnoho funkcionality pozmenil. Napríklad samotné vytvorenie virtuálu a prihlásenie sa do virtuálu pomocou vzdialenej plochy je zmenené, pretože táto možnosť nie je dohľadatelná na novom portáli. Pre sprístupnenie tejto funkcionality je potrebné zaslať požiadavok o jeho sprístupnenie. Starý portál pritom túto funkcionalitu ponúkal priamo v menu.



Obrázok 22: Windows Azure nový portál

3.5.3 Operačný systém pre vývoj a vývojové prostredia pre Windows Azure

Na samotný vývoj pre Windows Azure sa používa Visual Studio s doinštalovaným Azure SDK. Windows Azure projekty sa dajú spustiť v Azure virtuale, ktoré obsahuje doinštalované SDK. Na samotné nasadzovanie na Azure portál je potrebný X.509 v3 certifikát. Tento sa dá vytvoriť pomocou Visual Studio príkazového riadku.

> makecert -sky exchange -r -n "CN=<CertificateName>" -pe -a sha1 -len 2048 -ss My "<CertificateName>.cer"

Tento príkaz vytvorí certifikát, ktorý je potrebný uložiť na Azure portáli. Na novom portáli je ale problém s týmto certifikátom pretože ten podporuje len pfx certifikáty. Preto bolo treba skonvertovať tento certifikát a to nasledujúcim príkazom.

> makecert -sky exchange -r -n "CN=<CertificateName>" -pe -a sha1 -len 2048 -ss My "<CertificateName>.cer"

Posledná verzia Azure portálu podporuje obe verzie certifikátov. Potom je možné stiahnuť z portálu konfiguračný súbor do Visual Studia pre nasadenie na portál. Nasadenie trvá pár minút v závislosti na rozsahu projektu.

Azure ponúka zdarma najmenšiu inštanciu na tri mesiace. Po tomto čase prejde celý účet to módu, kde sa môže z databázy len čítať. Na jeho založenie je potrebný Live účet a kreditná karta, z ktorej sa stiahne euro, ktoré sa hneď vráti na účet a slúži len kvôli bezpečnostnému overeniu.

3.6 Porovnanie druhov notifikácii pre všetky platformy

Nasledujúca tabuľka zobrazuje rôzne druhy notifikácii pre rôzne platformy.

Android	iOS	Windows Phone	Windows 8
Normal notification	Alert	Toast	Toast
Big notification	Badge	Tile	Tile
Toast		Raw	Raw
			Badge
			Lock screen

Tabuľka 2: Porovnanie druhov notifikácii

Najmenej druhov má iOS, ktorý ako prvý použil notifikácie. Najviac druhov podporuje najnovší Windows 8. Ale väčší počet neznamená, že platforma ma aj väčšiu funkcionalitu v rámci notifikácii. Samotné druhy notifikácii sú rôzne obmedzené pre rôzne platformy, čomu sa budem venovať v implementačnej časti. Najväčšiu rôznorodosť notifikácii umožňuje Windows 8.

Jednotlivé názvy pre notifikácie nemajú rovnaký význam v iných platformách. Napríklad Toast notifikácia znamená v Androide jednoduchý oznam na spodnej strane obrazovky, čo odpovedá jednoduchému Alertu v iOS, vo Windows Phone oznam v hornej časti obrazovky a vo Windows 8 toast notifikácia znamená oznam v hornej pravej časti obrazovky ale aj s obrázkami.

Badge notifikácia v iOS a Windows 8 znamená zobraziť číslo na aplikácii, čo je súčasťou Normal notifikácie v Androide a Tile notifikácie vo Windows Phone. Android je chudobnejší v tomto ohľade, pretože tieto notifikácie sa zobrazujú len v notifikačnej lište a samotné ikony aplikácii sú statické.

Je zrejme z uvedených príkladov, že názvy a počet druhov notifikácii nie je rozhodujúci a ani možná funkcionalita, ktorej sa budem venovať pri implementácii jednotlivých notifikácii.

Základná architektúra je rovnaká pre všetky platformy. Samotná komunikácia so servermi je ale rozdielna z pohľadu zariadení aj samotných aplikačných serverov. Android zariadenie potrebuje na získanie kľúča pre dané zariadenie Google účet, no z ostatných zariadení tento účet potrebný nie je. Pre odoslanie požiadavku na notifikačný server Androidu je potrebný kľúč zariadenia a kľúč danej aplikácie zaregistrovanej na Google APIs. Pre odoslanie notifikácie na iOS je potrebný okrem ID aplikácie aj certifikát pridelený danej aplikácii pre push notifikácie, Windows Phone notifikácia potrebuje na svoje akceptovanie notifikačným serverom len ID zariadenia, alebo pokiaľ je aplikácia zabezpečená certifikátom pre notifikácie, tak týmto certifikátom. Windows 8 notifikácie ale už nie je možné posielať bez certifikátu, na základe ktorého získame prístupové kľúče na odoslanie požiadavku na notifikačný server WNS. Každá platforma má rozdielnu autentifikáciu, dokonca aj Windows Phone je rozdielny od Windows 8.

Vývojové prostredia a ich potrebné doplnky pre vývoj aplikácii s notifikáciami sú pre Android, Windows Phone a Windows 8 podobné. Stačí príslušný vývojársky nástroj a doplnok na vývoj mobilných aplikácii. Z tohto pohľadu je o čosi jednoduchšie nainštalovať Visual Studio a SDK 8, ako Eclipse, Javu, Android doplnok a aj balíček pre notifikácie. Vývoj pre iOS je problémový v tom, že treba mať MAC OS X. Na Windows Phone a Windows 8 je tiež potrebné Visual Studio so systémom Windows, ale tento systém je možné nainštalovať na akékoľvek zariadenie a je rozšírený.

4 Vývoj

Táto časť diplomovej práce sa zaoberá vývojom aplikácii pre jednotlivé platformy.

4.1 Vývoj aplikácie pre Android

Android je najrozšírenejším systémom pre mobilné zariadenie a táto kapitola popisuje implementáciu pre tento systém.

4.1.1 Popis klientskej aplikácie

Klientska aplikácia je vyvíjaná v prostredí Eclipse ako Android project, ktorý sa po doinštalovaní potrebných doplnkov zobrazí v menu s projektmi. Do projektu sme vložili súbor gcm.jar, ktorý je dostupný na stránkach [9], a umožňuje komunikovať s GCM serverom. Konfiguračný súbor AndroidManifest.xml treba nastaviť povolenia, aby podporoval notifikácie:

```
<permission android:name="app_package.permission.C2D_MESSAGE"
android:protectionLevel="signature" />
<uses-permission android:name="app_package.permission.C2D_MESSAGE" />
<uses-permission
android:name="com.google.android.c2dm.permission.RECEIVE" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
```

Zdrojový kód 1: Android manifest 1

A taktiež v tom istom súbore aj element application:

```
<receiver android:name="com.google.android.gcm.GCMBroadcastReceiver"
android:permission="com.google.android.c2dm.permission.SEND">
<intent-filter>
<action android:name="com.google.android.c2dm.intent.RECEIVE" />
<action android:name="com.google.android.c2dm.intent.REGISTRATION" />
<category android:name="app_package" />
</intent-filter>
</receiver>
<service android:name=".GCMIntentService" />
```

Zdrojový kód 2: Android manifest 2

Zaregistrovanie sa pomocou gcm.jar vykonáva nasledujúca metóda:

```
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      GCMRegistrar.checkDevice(this);
      GCMRegistrar.checkManifest(this);
      mRegistationId = GCMRegistrar.getRegistrationId(this);
      if (mRegistationId.equals("")) {
          GCMRegistrar.register(this, "Google Project ID");
      }
}
```

Zdrojový kód 3: Android registrácia GCM

Po získaní Device ID treba túto informáciu poslať na našu WCF servisu, ktorá ho potrebuje pre vytvorenie notifikácie. Android bez podpory tretej strany nepodporuje Soap volania. Preto sme museli sprístupniť vo WCF aj podporu pre REST volania, ktoré Android podporuje. Svoje Device ID posiela ako parameter:

```
HttpClient client = new DefaultHttpClient();
HttpGet request = new HttpGet();
request.setURI(new
URI("http://notificationandroid.cloudapp.net/Service1Andorid.svc/Subscri
beMyPhone?deviceId="+deviceId));
HttpResponse response = client.execute(request);
in = new BufferedReader
 (new InputStreamReader(response.getEntity().getContent()));
StringBuffer sb = new StringBuffer("");
String line = "";
String NL = System.getProperty("line.separator");
while ((line = in.readLine()) != null) {
sb.append(line + NL);
}
in.close();
String page = sb.toString();
```

Zdrojový kód 4: Android odoslanie ID

Azure WCF servisa si toto Device ID uloží do databáze. Jednou z najväčších výhod pri implementácii je to, že po prijatí notifikácie je volaná naša metóda, kde sa sami rozhodneme ako budeme notifikovať užívateľa a ako bude naša aplikácia reagovať na túto notifikáciu. Z tohto benefitu ale môžu vznikať problémy. Aplikácia má úplnú voľnosť ako zareaguje na notifikáciu a nič jej nebráni v tom aby začala sťahovať obrovské množstvo dát po sieti, alebo začala

vykonávať náročné operácie v systéme. Nasledujúca metóda vygeneruje samotnú notifikáciu po jej prijatí, aby upozornila užívateľa o novej udalosti:

```
protected void onMessage(Context context, Intent intent) {
      NotificationManager notificationManager = (NotificationManager)
      context.getSystemService(Context.NOTIFICATION_SERVICE);
      Notification notification = new Notification(icon, message, when);
      String title = context.getString(R.string.app_name);
      Intent notificationIntent = new Intent(context, Activity.class);
      notificationIntent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
      Intent.FLAG_ACTIVITY_SINGLE_TOP);
      PendingIntent intent =
      PendingIntent.getActivity(context, 0, notificationIntent, 0);
      notification.setLatestEventInfo(context, title, message, intent);
      notification.defaults |= Notification.DEFAULT_SOUND;
      notification.flags |= Notification.FLAG_AUTO_CANCEL;
      notificationManager.notify(0, notification);
}
```

Zdrojový kód 5: Android prijatie správy

Tento kód vytvorí Normal View notifikáciu, použije štandardný notifikačný tón a po tom ako na ňu užívateľ klikne sa táto notifikácia zavrie a spustí sa aplikácia zodpovedajúca za túto notifikáciu.

4.1.2 Inštalácia na emulátor a skutočné zariadenie

Pri spustení projektu Eclipse potrebuje nastaviť emulátor, pretože žiaden prednastavený neexistuje. Toto virtuálne zariadenie pridáme pomocou utility Android Virtual Device Manager v záložke Window. Jedno z hlavných nastavení je verzia operačného systému Android. Pre podporu notifikácii je potrebná verzia označená ako Google APIs. Potom už stačí zvoliť rozlíšenie obrazovky, prípadne veľkosť externej pamäte a názov zariadenia. Po dokončení sprievodcu nastavení emulátora sa toto zariadenie zobrazí medzi dostupnými emulátormi a pri spustení projektu ho môžeme zvoliť. Spustenie emulátoru trvá dlhšiu dobu ako emulátor pre Windows Phone alebo iOS. Po spustení sa musíme pripojiť pomocou Google Account v nastaveniach emulátoru, pretože bez aktívneho Google Account vráti GCM server chybu “RegistrationError”. Aplikácia sa spustí hneď po spustení a nainštalovaní do emulátora. To znamená, že pri prvom spustení novo vytvoreného emulátora, ktorý ešte nemá nadefinovaný Google Account, naša aplikácia nebude prijímať notifikácie. Po previazaní zariadenia s Google Account sa toto nastavenie zachová aj po vypnutí emulátora. Občas notifikácie prestali prichádzať, ktoré predtým fungovali, vtedy treba emulátor vypnúť a znova zapnúť. Reálne zariadenie tento problém nemá.

Obrázok 23: Android emulátor

Inštalácia na skutočné zariadenie je najjednoduchšia zo všetkých platforiem. Netreba mať nič doinštalované, stačí mať zapnuté nastavenie USB debugging na Android zariadení. Po pripojení zariadenia k počítaču sa toto zariadenie zobrazí medzi dostupnými zariadeniami pre spustenie. Skutočné zariadenie musí byť taktiež prepojené pomocou Google Account. Po určitom čase môže nastať problémy pri používaní toho istého Google Account pre emulátor aj skutočné zariadenie. Rovnako pri zmene hesla na Google Account, treba toto heslo zmeniť aj v telefóne, pretože notifikácie prestanú fungovať.

4.1.3 Popis serverovej aplikácie

Serverová aplikácia potrebuje len Device ID aby poslala notifikáciu na zariadenie cez GCM. Správy môžu mať tvar vo formáte JSON, alebo obyčajného textu. Nasledujúca tabuľka zobrazuje možnosti nastavenia správ.

Atribút	Popis
registration_id	String obsahujúci Device ID. Músí obsahovať minimálne 1 a maximálne 1000. Správa vo formáte obyčajného textu môže odosielať notifikáciu len na jedno zariadenie.
collapse_key	Toto nastavenie sa používa, ak je zariadenie offline a my chceme, aby po prechode do online stavu neprišli na zariadnie všetky správy počas nedostupnosti, ale len posledné.
Data	Telo notifikácie vo formáte kľúč a hodnota. Velkosť je obmedzená na 4Kb.
delay_while_idle	Nastavenie na true znamená, že GCM počká ak je

	zariadenie nečinné a notifikáciu pošle, až keď bude zariadenie aktívne. Pokiaľ nenastavíme tento parameter, použije sa hodnota false.
time_to_live	Hodnota je v sekundách a znamená ako dlho má byť notifikácia na GCM serveri, ak je zariadenie offline. Prednastavená hodnota sú 4 týždne.
restricted_package_name	String obsahujúci meno balíčku aplikácie. Pri odoslaní notifikácie, sa tieto notifikácie posielajú len na zariadenia s rovnakým názvom balíčku aplikácie.
dry_run	Nastavenie slúži na testovanie notifikácii, pretože sa notifikácia reálne neodošle na zariadenie, ale server ju spracuje.

Tabuľka 3: Možnosti Android notifikácii

Azure WCF servisa používa nasledujúcu metódu na odoslanie notifikácie:

```
WebRequest tRequest;
tRequest = WebRequest.Create("https://android.googleapis.com/gcm/send");
tRequest.Method = "post";
tRequest.ContentType = "application/x-www-form-urlencoded;charset=UTF8";
tRequest.Headers.Add(string.Format("Authorization: key={0}",
GoogleAppID));
String collaspeKey = Guid.NewGuid().ToString();
String postData =
string.Format("registration_id={0}&data.MyMessage={1}&collapse_key={2}&t
ime_to_live={3}&delay_while_idle={4}", item, message, collaspeKey,
2419200, false);
Byte[] byteArray = Encoding.UTF8.GetBytes(postData);
tRequest.ContentLength = byteArray.Length;
Stream dataStream = tRequest.GetRequestStream();
dataStream.Write(byteArray, 0, byteArray.Length);
dataStream.Close();
WebResponse tResponse = tRequest.GetResponse();
dataStream = tResponse.GetResponseStream();
StreamReader tReader = new StreamReader(dataStream);
String sResponseFromServer = tReader.ReadToEnd();
tReader.Close();
dataStream.Close();
tResponse.Close();
```

Zdrojový kód 6: Azure Android odoslanie

V response sa vráti kód 200, ak všetko prebehlo správne a priradene ID notifikácie. Aj v prípade chyby sa môže vrátiť hodnota 200, preto treba skontrolovať aj telo správy. Táto notifikácia sa pošle do konkrétneho zariadenia podľa URI.

4.1.4 EduKin aplikácia

EduKin využíva niekoľko rozmanitých aplikácii, ktoré pomáhajú vzdelávať, rehabilitovať, ale ponúkajú zábavu deťom trpiacim rôznymi poruchami učenia, koncentrácie, pohybu, stratou zraku alebo sluchu. Využíva technológie a zariadenia ako Kinect, Azure, Android a iné.

Časťou tejto diplomovej práce bolo aj pripraviť ukážkovú aplikáciu, ktorá bude využívať notifikácie posielané z Windows Azure na Android zariadenia. Samotná aplikácia posiela svoje získané Device ID na Azure server EduKinu, kde je pripravená metóda vychádzajúca z tejto práce, ktorá posiela notifikácie pri rôznych udalostiach.

Aplikácia prijíma notifikácie, ktoré ukladá a následne po spustení aplikácie všetky zobrazí v jednoduchom zozname. Pri vývoji nastal problém s volaním REST servisy, pretože od verzie Android 4, nie je možné volanie pomocou HttpClient v hlavnom vlákne. Preto treba po spustení aplikácie povoliť volanie nasledujúcim kódom, ktorý povoľuje toto volanie:

```
if (android.os.Build.VERSION.SDK_INT > 9) {
      StrictMode.ThreadPolicy policy = new
      StrictMode.ThreadPolicy.Builder().permitAll().build();
      StrictMode.setThreadPolicy(policy);
 }
```

Zdrojový kód 7: Android version policy

Ďalší problém bol s vytvorením inštalačného apk súboru. Od verzie Andoroid 4, musí byť aplikácia podpísaná. Na podpísanie je potrebné mať vytvorený Keystore v Eclipse. Ten je možne vytvoriť v príkazovom riadku Eclipse nasledujúcim kódom:

```
keytool -genkey -v -keystore MyAndroidKeys.keystore -alias
MyAndroidAlias -keyalg RSA -keysize 2048 -validity 10000
```

Zdrojový kód 8: Android vytvorenie keystore

V príkazovom riadku sa spustí sprievodca s dodatočnými nastaveniami, kde je potrebné zadať najmä heslo pre KeyStore.

Samotný export aplikácie do apk súboru je možné pomocou nástroja Export Android Application v menu Eclipse. Pri vytváraní použijeme vytvorený Keystore, kde po zadaní hesla sa nám vygeneruje podpísaný kľúč pre aplikáciu, ktorú je možné následne nainštalovať na skutočné zariadenie.

4.2 Vývoj aplikácie pre Windows 8

Nová verzia operačného systému Windows prináša notifikácie pre aplikácie prostredia Modern UI.

4.2.1 Popis klientskej aplikácie

Implementácia podpory notifikácii by mala byť jednoduchšia ako pre Windows Phone. Vo Visual Studio 2012 vyberieme novy projekt a zo zoznamu zvolíme Windows Store aplikáciu. Nastavíme aplikáciu na prijímanie notifikácii a to tak, že si zobrazíme Package.appxmanifest, kde je karta Application UI a sekcia notifications. V nej je potrebné povoliť toast notifikácie, pripadne zapnúť lock screen notifications. Pre tento typ notifikácii je podnebne použiť aj jeden z background tasks kontrolou: Control channel, Timer alebo Push Notification. Tie je možné pridať taktiež v Package.appxmanifest na záložke Available Declarations. Row notifikáciám sa budem venovať neskôr v tejto sekcii. Ostatné notifikácie sú zobrazovane systémom po prijatí z WNS automaticky. Aplikácia musí najskôr získať URI z tohto serveru. Na rozdiel od Windows Phone je to jednoduchšie. Získa ho nasledujúcim kódom:

```
public static PushNotificationChannel Channel { get; private set;}
private async void GetPushChannel ()
{
      channel = await
      PushNotificationChannelManager.CreatePushNotificationChannelForApp
      licationAsync();
}
```

Zdrojový kód 9: Windows 8 registrácia WNS

Volanie tejto metódy treba vložiť do spúšťacej metódy aplikácie. To garantuje, že URI je vždy aktuálna. Následne sa toto URI ktoré je prístupné cez propertu channel.Uri pošle serveru, ktorý ich bude odosielať, v tomto prípade je to Windows Azure.

Pre raw notifikácie je potrebne väčšie množstvo implementácie, ako ostatne druhy notifikácii. Najskôr je potrebne vytvoriť triedu, ktorá implementuje IBackgroundTask interface a musí referencovať Windows.ApplicationModel.background namespace. BackgroundTaskDeferral zaručuje aby proces náhodou neskočil chybou neočakávané, v rámci asynchrónneho volania, ako zobrazuje nasledujúci kód:

```
public async void Run(IBackgroundTaskInstance taskInstance)
{
      BackgroundTaskDeferral _deferral = taskInstance.GetDeferral();
      RawNotification notification =
      (RawNotification)taskInstance.TriggerDetails;
      string content = notification.Content;
      var result = await ExampleRawMethodAsync();
      _deferral.Complete();
}
```

Zdrojový kód 10: Windows 8 Background Task

Uvedený background task musí byť zaregistrovaný v manifeste aplikácie. Je to možné cez designer na karte Declarations, kde z Available Declarations vyberieme Background Tasks a následné označíme checkbox Push notification a v poli Start page vypíšeme našu triedu, ktorá implementuje požadovaný interface. Treba si dať pozor na registráciu background tasks, aby neboli zaregistrované viac násobne, pretože by vyťažovali procesor. Takto vytvorená notifikácia spustí kód na pozadí.
Push notifikácie je možné pridávať aj do Lock screen obrazovky Windows 8 a je potrebné potvrdenie od užívateľa pre ich fungovanie. To získame nasledujúcim kódom:

```
BackgroundAccessStatus status =
BackgroundExecutionManager.GetAccessStatus();
if (status == BackgroundAccessStatus.Unspecified)
await BackgroundExecutionManager.RequestAccessAsync();
```

Zdrojový kód 11: Windows 8 potvrdenie lock screen

4.2.2 Inštalácia na emulátor a skutočné zariadenie

Samotná inštalácia na zariadenie je jednoduchá, keďže je to zariadenie na ktorom sa vyvíja, teda Windows 8. S prihláseným developerským účtom je priamo vo Visual Studio možnosť spustiť aplikáciu na Lokal machine. Je tam možnosť aj spustenia v emulátore, tato možnosť spusti aktuálne bežiaci systém v menšom okne virtuálneho tabletu, kde sa dajú rýchlo nastavovať rozlíšenia obrazovky, natočenie a ďalšie vlastnosti pre ktoré treba aplikácie optimalizovať.

Obrázok 24: Windows 8 emulátor

Po spustení aplikácie cez Visual Studio na local machine, sa táto aplikácia nainštaluje na počítač, pridá sa automaticky dlaždica na štartovaciu obrazovku a tato aplikácia sa spustí. Funkcionalita napríklad pre debugovanie je rovnaká ako pre vývoj klasickej Windows form aplikácie.

4.2.3 Popis serverovej aplikácie

Serverová aplikácia je nezávislá na použitej platforme, pretože WNS používa na komunikáciu štandardne protokoly REST alebo JSON. Na odosielanie notifikácii je na Azure nasadená WCF servisa. Server sa musí najskôr autenifikovať voči WNS. Potrebuje na to SID a SecretKey, ktoré získame z portálu pre Windows Store app development založením novej aplikácie. Po par krokoch sa zobrazí SID a SecretKey v záložke Authenticating your service. Samozrejme na založenie aplikácie je potrebný developersky účet Microsoftu. Následne tieto dáta pošle cez HTTPS request na WNS, toto vykoná nasledujúca metóda:

```
protected OAuthToken GetAccessToken(string secret, string sid)
{
      var urlEncodedSecret = HttpUtility.UrlEncode(secret);
      var urlEncodedSid = HttpUtility.UrlEncode(sid);
      var body =
      String.Format("grant_type=client_credentials&client_id={0}&client_
      secret={1}&scope=notify.windows.com", urlEncodedSid,
      urlEncodedSecret);
      string response;
      using (var client = new WebClient())
      {
```

```
            client.Headers.Add("Content-Type", "application/x-www-form-
            urlencoded");
            response =
            client.UploadString("https://login.live.com/accesstoken.srf"
            , body);
      }
}
```

Zdrojový kód 12: Windows 8 odoslanie na WNS

Metóda získa token, ktorý je potrebné previesť do nasledujúceho formátu:

```
public class AuthToken
{
      [DataMember(Name = "access_token")]
      public string AccessToken { get; set; }
      [DataMember(Name = "token_type")]
      public string TokenType { get; set; }
}
```

Zdrojový kód 13: Windows 8 Token

Tento token ale WNS vracia vo formáte JSON, ktorý je potrebné transformovať na štruktúru tokenu, ktorý sme zadefinovali. O to sa stará táto metóda:

```
private void GetAuthTokenFromJson(string jsonString)
{
      using (var ms = new
      MemoryStream(Encoding.Unicode.GetBytes(jsonString)))
            {
            var ser = new DataContractJsonSerializer(typeof(AuthToken));
            var AuthToken = (AuthToken)ser.ReadObject(ms);
            }
}
```

Zdrojový kód 14: Windows 8 získanie Tokenu

Po získaní autentifikačného kľúču stačí už len poslať samotnú notifikáciu so všetkými potrebnými údajmi vo formáte buď JSON alebo XML na WNS server. Nasledujúca tabuľka popisuje štruktúru XML, ktoré sa posiela do metódy PostToWns().

Notific	Notification	Notification Data	Content-Type

ation Name	Type		header
Badge	"wns/badge"	"<?xml version='1.0' encoding='utf-8'?><badge value=\"2\"/>"	"text/xml"
Tile	"wns/tile"	"<?xml version='1.0' encoding='utf-8'?><tile><visual version=\"1\"><binding template=\"TileWideText03\"><text id=\"1\">Hello World! My tile notification</text></binding><binding template=\"TileSquareText04\"><text id=\"1\">Hello World!</text></binding></visual></tile>"	"text/xml"
Toast	"wns/toast"	"<?xml version='1.0' encoding='utf-8'?><toast><visual version='1'><binding template='ToastText01'><text id='1'>My first Toast</text></binding></visual></toast>"	"text/xml"
Raw	"wns/raw"	"{notification: raw, text: 'Raw notification string'}"	"application/octet-stream"

Tabuľka 4: Windows 8 štruktúra notifikácii

Vytvorené XML sa pošle ako parameter do metódy, ktorá ho použije spolu so získanými prístupovými údajmi a vytvorí a pošle samotnú notifikáciu na WNS:

```
public string PostToWns(string secret, string sid, string uri, string
xml, string notificationType, string contentType)
{
      var accessToken = GetAccessToken(secret, sid);
      byte[] contentInBytes = Encoding.UTF8.GetBytes(xml);
      HttpWebRequest request = HttpWebRequest.Create(uri) as
      HttpWebRequest;
      request.Method = "POST";
      request.Headers.Add("X-WNS-Type", notificationType);
      request.ContentType = contentType;
      request.Headers.Add("Authorization", String.Format("Bearer {0}",
      accessToken.AccessToken));
      using (Stream requestStream = request.GetRequestStream())
```

```
        requestStream.Write(contentInBytes, 0, contentInBytes.Length);
        using (HttpWebResponse webResponse =
        (HttpWebResponse)request.GetResponse())
        return webResponse.StatusCode.ToString();
}
```

Zdrojový kód 15: Windows 8 odoslanie notifikácie na WNS

V response sa vráti kód 200 ak všetko prebehlo správne. Tato notifikácia sa pošle do konkrétneho zariadenia podľa URI. Samozrejme toto zariadenie musí mat online pripojenie, inak notifikácia podľa nastavenia príde až po pripojení sa do online siete.

4.3 Vývoj aplikácie Windows Phone 7 a Windows Phone 8

Windows Phone notifikácie sú dlhšie používané ako pre Windows 8, preto je aj ich implementácia náročnejšia, ako sa popisuje v nasledujúcej kapitole.

4.3.1 Popis klientskej aplikácie

Aplikácie vytvorené pre Windows Phone 7 sú spustiteľné aj na Windows Phone 8. Vo Visual Studiu vytvoríme Windows Phone projekt s podporou verzie systému od verzie 7. Podporu notifikácii nastavíme vo vlastnostiach Manifest.xml súboru. Na záložke Capabilities zapneme možnosť ID_CAP_PUSH_NOTIFICATION. Toto nastavenie zaručuje, že aplikácia bude prijímať notifikácie. Na notifikácie potrebujeme získať URI. Toto získame nasledujúcim kódom:

```
Guid deviceId = Guid.NewGuid();
HttpNotificationChannel myPushChannel=
myPushChannel = HttpNotificationChannel.Find("myChannel");
if (myPushChannel == null)
{
        myPushChannel = new HttpNotificationChannel("myChannel");
        myClient.SubscribeMyPhoneAsync(deviceId, myPushChannel.ChannelUri.
        ToString());
        myPushChannel.Open();
}
else
{
        myClient.SubscribeMyPhoneAsync(deviceId, myPushChannel.ChannelUri.
        ToString());
}
```

Zdrojový kód 16: Windows Phone registrácia MPNS

URI sa nachádza v premennej myPushChannel, v jej ChannelUri. Pre podporu Toast a Tile notifikácii je potrebné ich povolenie v aplikácii pre získaný HttpNotificationChannel, pri každom jeho získavaní, čo vykonáva nasledujúci kód:

```
if (myPushChannel.IsShellTileBound == false)
{
      var ListOfAllowedDomains = new Collection<Uri> { new Uri(@"http://
      notificationWindowsPhone.cloudapp.net") };
      myPushChannel.BindToShellTile(ListOfAllowedDomains);
}
if (myPushChannel.IsShellToastBound == false)
{
      myPushChannel.BindToShellToast();
}
```

Zdrojový kód 17: Windows Phone Tile obrázky

Kód okrem povolenia Toast a Tile notifikácii nastavuje domény, z ktorých je možné prijímať obrázky, ktoré sa zobrazujú na dlaždiciach v našom prípade je to adresa notifikačného serveru, kde sú obrázky uložené. Tile notifikácie, ktoré zároveň posielajú v notifikácii aj adresu obrázku, ale nefungujú v školskej wifi sieti. Notifikácia je doručená a zobrazí sa podľa nastavení, ale obrázok sa na dlaždici neaktualizuje. V inom internetovom pripojení napríklad od Telecomu alebo UPS sa notifikácia zobrazí aj so správnym obrázkom.

Pre fungovanie Raw notifikácii, ktoré zobrazujú informácie počas toho ako je aplikácia spustená, je potrebná dodatočná implementácia. Po získaní URI je potrebné zaregistrovať metódu, ktorá zobrazí tento typ notifikácie.

```
myPushChannel.HttpNotificationReceived += new EventHandler<HttpNotificat
ionEventArgs>(myPushChannel_HttpNotificationReceived);
 void myPushChannel_HttpNotificationReceived(object sender, HttpNotifica
tionEventArgs e)
{
      Deployment.Current.Dispatcher.BeginInvoke(() =>
{
      StreamReader myReader = new StreamReader(e.Notification.Body);
      String
      rawNotification = "Received Raw Notification: " + myReader.ReadToE
      nd();
});
}
```

Zdrojový kód 18: Windows Phone Raw notifikácie

Prijatá raw notifikácia sa uloží do premennej rawNotification.

4.3.2 Inštalácia na emulátor a skutočné zariadenie

Spustenie v emulátore je jednoduché. Stačí nastaviť spustenie v emulátore a projekt vo Visual Studiu spustiť. Po pár sekundách sa zobrazí emulátor, do ktorého sa nainštaluje naša aplikácia. Pre podporu Tile notifikácii je potrebné túto aplikáciu zobraziť na štartovacej obrazovke systému.

Obrázok 25: Windows Phone emulátor

Problém nastal s Tile typom notifikácii. Pri pokuse o pridanie počtu nových správ, čo zobrazuje číslo umiestnené v pravom hornom rohu, sa na bielom pozadí zobrazilo biele číslo. Po mnohých neúspešných pokusoch tento problém odstrániť rôznymi nastaveniami, pomohla až aktualizácia celého Visual Studia.

Inštalácia na skutočné zariadenie je taktiež rozdielna pre obe verzie systému Windos Phone. Prvá veria Windows Phone 7 potrebuje mať v počítači nainštalovaný software od Microsoftu nazvaný Zune. Po jeho nainštalovaní je možne nainštalovať aplikáciu na Windows Phone 7 zariadenie pripojené k tomuto počítaču. Pre Windows Phone 8 je nutné mať nainštalovaný Windows 8 a SDK 8 a žiaden dodatočný software nie je potrebný. Na SDK 8 je ale potrebné nové Visual Studio 2012, v ktorom sa následne po pripojení zariadenia zobrazí medzi dostupnými zariadeniami na spustenie. Samotný emulátor sa správa identicky ako skutočné zariadenia.

4.3.3 Popis serverovej aplikácie

MPNS používa na komunikáciu štandardne protokoly REST alebo JSON. Na odosielanie notifikácii je na Azure nasadená WCF servisa. Tá sa nemusí autentifikovať ako je to u Windows 8,

pokiaľ aplikácia nepoužíva certifikát, ale stačí jej URI zariadenia. Z tohto URI sa vytvorí v C# web request:

```
var myRequest = (HttpWebRequest)WebRequest.Create(“URI”);
myRequest.Method = WebRequestMethods.Http.Post;
myRequest.ContentType = "text/xml";
myRequest.ContentLength = pushPayload.Length;
myRequest.Headers.Add("X-MessageID", Guid.NewGuid().ToString());
```

Zdrojový kód 19: Windows Phone hlavička requestu

Do hlavičky requestu sa pridá druh notifikácie, teda Toast, Tile alebo Raw notifikácia:

```
String notificationType = “”;
switch (notificationType)
{
      case "toast":
      myRequest.Headers["X-WindowsPhone-Target"] = "toast";
      myRequest.Headers.Add("X-NotificationClass",”2”);
      break;
      case "tile":
      myRequest.Headers["X-WindowsPhone-Target"] = "token";
      myRequest.Headers.Add("X-NotificationClass", “1”);
      break;
      case "raw":
      myRequest.Headers.Add("X-NotificationClass", ”3”);
      break;
}
```

Zdrojový kód 20: Windows Phone xml notifikácii

A na koniec sa pridá samotný typ notifikácie aj s nastaveniami danej notifikácie v xml tvare. Nasledujúca tabuľka popisuje štruktúru XML, ktoré sa posiela na notifikačný server MPNS do metódy PostToWns().

Notification Name	Notification Type	Notification Data	Content-Type header
Tile	"wp/tile"	"<?xml version='1.0' encoding='utf-8'?><tile><text id=\"1\"> My tile notification </text</tile>"	"text/xml"

Toast	"wp/toast"	"<?xml version='1.0' encoding='utf-8'?><toast><text id='1'>My first Toast</text></binding></visual></toast>"	"text/xml"
Raw	"wp/raw"		"text/xml"

Tabuľka 5: Windows Phone štruktúra notifikácii

V tele raw notifikácie môže byť čokoľvek. V response sa vráti kód 200 ak všetko prebehlo správne. Táto notifikácia sa pošle do konkrétneho zariadenia podľa URI. Samozrejme toto zariadenie musí mat online pripojenie, inak notifikácia podľa nastavenia príde až po pripojení sa do online siete.

4.4 Vývoj aplikácie pre iOS

iOS sa vyvíja v jazyku Objective C a nasledujúca kapitola sa venuje implementácii aplikácie v tomto jazyku.

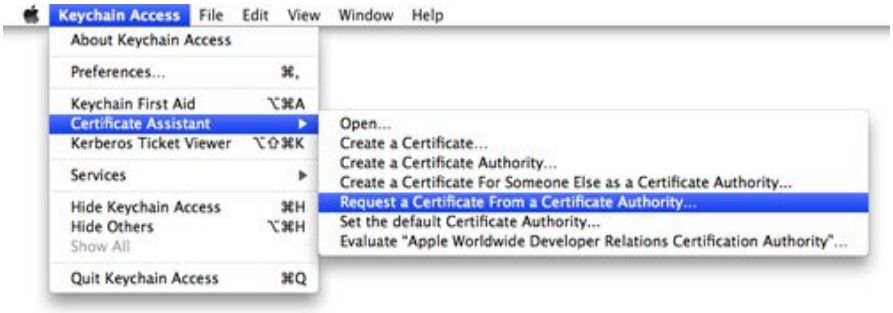
4.4.1 Popis klientskej aplikácie

Aplikácie pre iOS sa vyvíjajú v programe Xcode. Po spustení tohto vývojového nástroja, založíme projekt a následne vyberieme iOS Aplication. Zariadenie potrebuje získať token z APNS serveru. Na to je potrebné pridať premennú do vygenerovaného súboru AppDelegate.h ako zobrazuje nasledujúci kód:

```
@property (strong, nonatomic) NSString *deviceToken;
Zaregistrovanie sa vykonáva vo vygenerovanom súbore AppDelegate.m, kde
prepíšeme existujúci handler didFinishLaunchingWithOptions:
-(BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:
(NSDictionary *)launchOptions
{
[[UIApplication sharedApplication] registerForRemoteNotificationTypes:
UIRemoteNotificationTypeAlert | UIRemoteNotificationTypeBadge |
UIRemoteNotificationTypeSound];
eturn YES;
}
```

Zdrojový kód 21: iOS registrácia APNS

Pre fungovanie notifikácii je potrebný certifikát, ktorý je potrebný nielen pre aplikáciu ale aj pre odosielanie notifikácii z Azure serveru. Tento certifikát získame z programu Keychain Access na záložke Certificate Assistant ako zobrazuje obrázok:

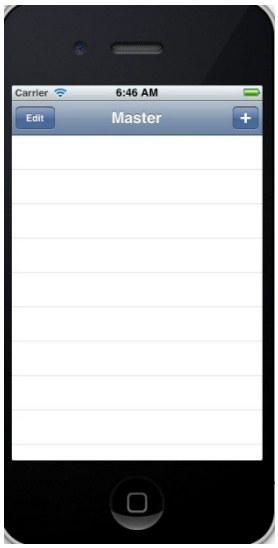


Obrázok 26: iOS vytvorenie certifikátu

Na tento certifikát je potrebný developerský účet. Tento certifikát nemáme, preto táto aplikácia nemôže prijímať push notifikácie.

4.4.2 Inštalácia na emulátor a skutočné zariadenie

Spustenie aplikácie z Xcode, spustí emulátor, do ktorého sa nainštaluje a spustí naša aplikácia bez potrebného dodatočného nastavovania. Aplikácia sa spustí takmer okamžite po spustení emulátora. Po zastavení aplikácie v Xcode, ostáva tento emulátor spustený na úvodnej obrazovke systému podobne ako emulátory ostatných platforiem.



Obrázok 27: iOS emulátor

Pre inštaláciu na skutočnom zariadení je potrebný certifikát. Apple Developerský program umožňuje používať až 100 zariadení na testovacie účely s použitím potrebného certifikátu, ktorý ale nemáme. Tento certifikát je potrebné prepojiť s konkrétnou aplikáciou.

4.4.3 Popis serverovej aplikácie

Odosielanie push notifikácii na APNS server uľahčuje použitie projektu dostupného na portáli [10]. Projekt má meno apns.net a umožňuje jednoducho posielať notifikácie na APNS server. Nasledujúca ukážka kódu používa triedu z tohto programu nazvanú ApnsNotificationChannel pre vytvorenie notifikácie:

```
X509Certificate certificate = new X509Certificate(@"certificatPath");
using (ApnsNotificationChannel channel =
ApnsNotificationChannel.OpenNotificationChannel(certificate, true))
{
DeviceToken deviceToken = "";
NotificationPayload payload = new NotificationPayload("Message");
ApnsNotification notification = new ApnsNotification(deviceToken,
payload);
channel.SendNotification(notification);
var results = channel.Close();
}
```

Zdrojový kód 22: iOS serverová aplikácia

Pre odoslanie notifikácie potrebujeme mať na servery nainštalovaný certifikát získaný a Apple Developerského programu, ktorý je prepojený s našou aplikáciou. Tento certifikát ale nemáme. A keďže ho nemáme, naša iOS aplikácia nezíska token z APNS serveru, ktorý je potrebný na identifikáciu zariadenia, na ktoré sa má notifikácia poslať.

4.5 Porovnanie, spoločné vlastnosti, hlavne rozdiely v implementácii

Implementácia notifikácii pre mobilné aplikácie je pre Windows Phone 7, Windows Phone 8, Windows 8 a iOS podobná, pretože prijaté notifikácie sú naimplementované v systéme, ktorý sa sám stará o ich obsluhu. Pre vývoj na Android je potrebné tuto notifikáciu po prijatí naprogramovať, čo pridáva veľkú voľnosť programátorom v obsluhovaní notifikácii, no implementácia je náročnejšia a ako už v tejto diplomovej práci bolo spomenuté, môže výrazne spomaľovať samotný systém pri nesprávnej implementácii.

Používanie emulátorov je opäť veľmi podobné pre Windows Phone 7, Windows Phone 8, Windows 8 a iOS, pretože sa spúšťa bez nutnej konfigurácie. Naopak emulátor pre Android treba vytvoriť a nastaviť aby bolo možné aplikáciu spustiť.

Spustenie aplikácie je veľmi jednoduché pre Android platformu, keďže stačí zariadenie len pripojiť a povoliť USB Debugging. Windows Phone 7 potrebuje nainštalovaný software Zune, Windows Phone 8 potrebuje nainštalovaný Windows 8 a SDK 8, rovnako ako Windows 8.

Inštalácia na reálny iPhone je najviac zložitý zo všetkých spomínaných platforiem, pretože potrebuje špeciálny certifikát pre aplikáciu aby mohla byť testovaná a spustená na konečnom počte zariadení.

Komunikáciu so serverom obmedzoval len Android, pretože nepodporuje Soap volanie, bez aplikácie tretej strany. To ostatné platformy podporujú.

Architektúra serverovej časti je ale veľmi podobná, pretože okrem autentifikačných údajov sa posielajú na server požiadavky v určitom formáte. Následne sa o spracovanie a odoslanie notifikácii starajú samotné notifikačné servery platforiem.

5 Windows Azure

Pri vývoji v prostredí Windows Azure musíme počítať s tým, že Microsoft môže kedykoľvek aktualizovať alebo dokonca nasadiť náš projekt na iný virtual s rovnakým systémom. Na druhú stranu nám zjednodušuje nasadenie, správu systému, konfiguráciu. Azure ponúka tri hlavné typy vývoja:

- Web role ASP.NET alebo WCF
- Worker role
- Virtual Machine (VM) role

Prioritou pre web role je hosťovanie webových stránok pod plným IIS (Internet Information Service). Môže hosťovať viacero stránok pod jednou web role. Worker roles sú určené na dlho bežiace aplikácie plniace funkciu podobnú Windows service. Jediný veľký rozdiel medzi web role a worker role je ten, že web role hosťuje IIS. VM role umožňuje nahrať vlastný operačný systém napríklad Windows Server alebo Linux. Je určený pre riešenia, ktoré sú príliš komplexné, aby mohli využívať prvé dva typy. Takto vytvorený systém ale musíme spravovať a konfigurovať a vývoj aplikácii sa značne predĺži. Microsoft odporúča jednu z prvých dvoch možností s použitím startup tasks ako je uvedené v [8], ktorým sa ešte budem venovať neskôr. Preto som aj ja zvolil prvú možnosť, teda web role, konkrétne WCF, ktorá sa nasadí na Windows Server.

Môžeme si vybrať operačný systém, na ktorý chceme nasadzovať náš projekt. Na výber je viacero systémov a pri písaní tejto diplomovej práce bol zoznam nasledujúci:

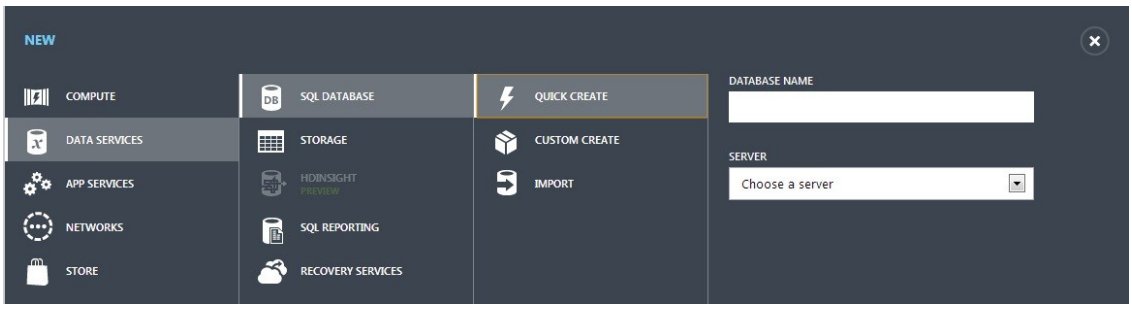
- Microsoft BiztalkServer 2013 Beta
- Microsoft SQL Server 2012 Eval Edition
- Windows Server 2008 R2 SP1
- Windows Server 2012
- OpenLogic CentOS 6.3
- SUSE Linux Enterprise Server 11 SP2
- Ubuntu Server 12.04.1 LTS
- Ubuntu Server 12.10

Zariadenia, na ktoré budeme posielať notifikácie musia svoje prístupové údaje ukladať niekde v našom projekte. Pre ukladanie dát ponúka Azure tieto možnosti:

- Blobs
- Tables
- Queues
- Local storage
- Azure Drive
- SQL Azure

Azure Blobs sa používajú , pokiaľ chceme ukladať veľké dáta. Tables sú finančne efektívne riešenie podobné SQL tabuľkám, ale s obmedzeniami, napríklad na množstvo dotazov. Local storage a Azure Drive je riešenie podobné NTFS disku. Azure queues sú vhodné pre nepripojené prostredia. To znamená, že úloha na spracovanie sa vloží do Azure queues, ale o spracovaní tejto úlohy sa už užívateľ nedozvie. Úlohu spracováva jeden alebo viac robotov, ktorý ju po úspešnom spracovaní odstránia z fronty, alebo ju vo fronte ponechá, pokiaľ nastane nejaký problém pri jej spracovaní. Problémom ale pri takomto spôsobe spracovania je, že jedna úloha, ktorá spôsobí nejakú chybu, spôsobí zastavenie celého systému, pretože ju nebude vedieť spracovať žiaden robot. Takéto správy sú označované ako poison messages a na takéto správy musí byť naimplementovaná dodatočná funkcionalita, ktorá bude takéto správy odhaľovať. Spravidla sa tento spôsob ukladania dát uplatní najmä pre náročné úlohy na spracovanie, čo ale notifikácie nie sú, preto som zvolil posledný typ a to SQL Azure.

SQL Azure je relačná databáza, obdoba Microsoft SQL databáze v cloude. Vytvára sa na Azure portáli pod záložkou SQL Databases ako je zobrazené na nasledujúcom obrázku.



Obrázok 28: Windows Azure SQL databáza

Na spravovanie Azure SQL serveru existuje aj jednoduché webové rozhranie, kde je možné vytvárať tabuľky, alebo sa dotazovať pomocou sql príkazov. Tento diplomový projekt využíva Entity Framework pre jeho jednoduché používanie a rozšíriteľnosť. Vytvorený je pomocou Visual Studia a vygenerovaný sql script sme spustili práve pomocou spomínaného webového rozhrania. Do tabuliek sa ukladajú zaregistrované zariadenia, na ktoré sa posielajú notifikácie. V prípade, že nejaká notifikácia nebude odoslaná kvôli nejakému problému, je teda podobná spomínaným poison messages pre Azure queues, tento problém zapisujem do logov, ktoré nie sú zmazané ani po preinštalovaní systému.

Doménové meno pre službu zvolíme pri vytváraní služby na Azure portále. Pod týmto názvom bude dostupná z internetu. Má tvar “servicename.cloudapp.net”, to znamená, že všetky názvy sú pod doménou “cloudapp.net”. DNS systém podporuje taktiež CNAME záznamy, ktoré mapujú jednu doménu do ďalšej.

Dôležitou časťou Windows Azure je diagnostika. Nielen z dôvodu prípadnej identifikácie problému zo spustenej aplikácie, ale aj kvôli vyťaženosti serveru a jeho prípadnému

aktualizovaniu o väčší výpočetný výkon alebo kapacitu pre dáta. Konfigurácia sa vykonáva na úrovni inštancie. Widnows Azure Diagnostics podporuje nasledujúce typy diagnostiky:

- Windows Azure basic logs
- Performance counters
- Windows Event Logs
- Windows Azure Diagnostic infrastructure logs

Windows Azure basic logs zaznamenávajú informácie zapísané do Windows Azure trace listener. Performance counters zaznamenávajú informácie z nakonfigurovaných performance counters. Takto je možné sledovať vyťaženie napríklad procesora, alebo pamäte. Windows Event Logs zbierajú dáta z nakonfigurovaných Event Logs. Windows Azure Diagnostic infrastructure logs zaznamenávajú data, ktoré produkujé nakonfigurovaný Windows Azure Diagnotics process.

Windows Azure Diagnostics podporuje nasledujúce úložiská pre diagnostiku:

- IIS Logs
- IIS Failed Request Logs
- Crash dumps
- Custom directories

IIS logs, IIS Failed Request Logs a Crash dumps sa evidujú samy. Pri Custom directories Windows Azure podporuje asociáciu s ktorýmkoľvek adresárom v inštancii. Táto možnosť preto ponúka aj integráciu s logovacími službami tretích strán do Windows Azure Diagnostics.

Táto diplomová práca využíva Windows Azure basic logs, teda Windows Azure trace listener. Pre toto logovanie je potrebné nastaviť web.config web role. Nasledujúci ukážka kódu [] pridáva trace listener do kolekcie trace listeners pre web role.

```
<system.diagnostics>
      <trace>
              <listeners>
                  <add
                  type="Microsoft.WindowsAzure.Diagnostics.DiagnosticMoni
                  torTraceListener, Microsoft.WindowsAzure.Diagnostics,
                  Version=1.8.0.0,Culture=neutral,
                  PublicKeyToken=31bf3856ad364e35"
                  name="AzureDiagnostics">
                        <filter type="" />
                  </add>
            </listeners>
      </trace>
```

```
<-system.diagnostics>
```

Zdrojový kód 23: Windows Azure logs 1

Následne vo webrole.cs triede nastavíme kam sa majú ukladať dáta, ako často a v akých množstvách, čo zobrazuje nasledujúci kód:

```
public override bool OnStart()
{
      var diagnostics =
      DiagnosticMonitor.GetDefaultInitialConfiguration();
      diagnostics.Logs.ScheduledTransferLogLevelFilter =
      LogLevel.Undefined;
      diagnostics.Logs.ScheduledTransferPeriod =
      TimeSpan.FromMinutes(1);
      diagnostics.Logs.BufferQuotaInMB = 10;
 CloudStorageAccount account =
 CloudStorageAccount.Parse(RoleEnvironment.GetConfigurationSettingValue(
 "Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString"));
      DiagnosticMonitor.Start(account, diagnostics);
      return base.OnStart();
}
```

Zdrojový kód 24: Windows Azure logs 2

V konfiguračnom súbore nastavíme miesto pre logy:

```
<ConfigurationSettings>
      <Setting
      name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString"
      value="DefaultEndpointsProtocol=https;AccountName=;AccountKey=" />
</ConfigurationSettings>
```

Zdrojový kód 25: Windows Azure logs 3

V kóde aplikácie môžeme následne zapisovať informácie do logu. Nasledujúci kód zobrazuje použitie tejto konfigurácie pre zápis informácii, upozornení alebo chybných správ:

```
System.Diagnostics.Trace.TraceInformation("Information");
System.Diagnostics.Trace.Warning("Warning ");
System.Diagnostics.Trace.TraceError("Error");
```

Zdrojový kód 26: Windows Azure logs 4

Zaznamenané informácie je možné zobrazovať, napríklad pomocou Diagnostics Agent, alebo je možné tieto informácie spracovávať a triediť ich pomocou kódu.

Použitím web role sa zjednodušuje administrácia. Správu systému má na zodpovednosti Microsoft a tým vznikajú problémy spojené s vlastnou administráciou alebo inštaláciou nových programov. O všetky nastavenia a inštalácie prídeme v momente preinštalovania systému, ktorému sa nemôžeme vyhnúť. Riešením tohto problému sú Startup tasks. Je to script, ktorý sa vykonáva vždy po štarte inštancie. Tento script musí byť robustný voči chybám, pretože pri pád scriptu spôsobí reštart inštancie. Existujú tri typy spustenia startup task:

- Simple
- Background
- Foreground

Simple znamená, že systém musí počkať, kým script skončí. Pri background nastavení sa script vykonáva na pozadí a systém pokračuje s nainštalovaním napríklad web role. Tento typ je využiteľný pri dlho bežiacich startup script. Posledné z možných nastavení je foreground, ten je podobný background s tým rozdielom, že inštancia sa nezrecykluje počas vykonávania startup script. Pre posielanie notifikácii je potrebný prvý typ, pretože napríklad notifikácie pre iOS potrebujú nainštalovaný certifikát a až po nainštalovaní sa môžu volať notifikácie. Spustenie sa nastavuje v service definition súbore našej web role nasledujúcim ako je v ukážke kódu [5].

```
<Startup>
      <Task commandLine="Startup.cmd" executionContext="elevated"
      taskType="simple" />
</Startup>
```

Zdrojový kód 27: Windows Azure Starup task

Samotný súbor Startup.cmd môže spúšťať PowerShell script. Takto môžeme napríklad aj nainštalovať certifikát potrebný pre notifikácie.

5.1 Práca s notifikačnými servermi

Nasledujúce tabuľky zobrazujú možné odpovede a ich popis z notifikačných serverov. Odpovede sú potrebné pre správne identifikovanie chyby pri pokuse o odoslanie notifikácie na koncové zariadenie a následne ďalší postup pri jej odosielaní.

Google Cloud Messaging:

HTTP Response	Popis

200	Správa bola spracovaná úspešne. Telo response obsahuje detailnú správu. Táto správa môže obsahovať: • multicast_id – unikátne ID identifikujúce správu • success – počet správ, ktoré boli úspešne spracované • failure – počet správ, ktoré neboli úspešne spracované • cannonical_ids – registračné ID zariadenia, pokiaľ aplikácia vyvolala viacero registrácii pre jedno zariadenie. Cannonical_ids je posledné registračné ID zariadenia pri jeho registrácii • results – pole statusov o notifikácii o message_id – string reprezentujúci správu, ktorá bola úspešne spracovaná o registration_id – znamená, že GCM spracoval správu ale tá má ďalšie cannonical_id pre dané zariadenie o error – string popisujúci vzniknutú chybu (MissingRegistration, InvalidRegistration, MismatchSenderId, NotRegistered, MessageTooBig, InvalidDataKey, InvalidTtl)
400	Použiteľný len pre JSON request. Indikuje, že request sa nemá správny formát.
401	Chyba, ktorá indikuje, že API key nie je validný.
5xx	Chybné správy v rozsahu 500-599, informujú, že sa vyskytla interná chyba v GCM počas procesu spracovania a odosielania notifikácie. Požiadavok musíme zopakovať neskôr.

Tabuľka 6: GCM response

Microsoft Push Notification Service:

HTTP Response	Popis
200	Správa bola spracovaná úspešne a pridaná do zoznamu na odoslanie a status obsahuje jedno z nasledujúcich: Received, QueueFull, Suppressed.
400	Tento response sa pošle, pokiaľ náš server pošle správu so zle naformátovanému XML alebo URI zariadenia.
401	Poslaná notifikácia nie je autorizovaná, napríklad kvôli nesprávnemu certifikátu.
404	Predplatné je neplatné a nie je prítomné na službe pre push notifikácie.
405	Nesprávna metóda (PUT, DELETE, CREATE). Len POST je povolená, keď sa posiela push notifikácia.

406	Táto odpoveď je vrátená v prípade, ak neautorizovaný cloud server vyčerpá denný limit pre notifikácie. O znovu odoslanie notifikácie by sa mal pokúsiť po 24 hodinách, keď bude notifikačný cyklus znova pokračovať.
412	Zariadenie je v offline režime. Pre cloud server to neznamená znovu odoslanie notifikácie, pretože, keď bude zariadenie online, MNPS bude pokračovať v spracovaní notifikácie a tá mu bude doručená.
503	Push Notification Service nie je schopná spracovať request. Tento request by mal byť odoslaný znova neskôr.

Tabuľka 7: MPNS response

Windows Notification Services:

HTTP Response	Popis
200	Notifikácia bola akceptovaná WNS serverom.
400	Hlavička requestu nebo zadaná správne.
401	Poslaná notifikácia nie je autorizovaná, napríklad kvôli nesprávnemu, respektíve nezadanému tokenu.
403	Prístupový token z requestu sa nezhoduje s prihlasovacími údajmi aplikácie (SecretKey, SID) z Microsoft Dashboard.
404	URI nie je validný alebo nie je rozpoznaný serverom WNS.
405	Nesprávna metóda (PUT, DELETE, CREATE). Len POST je povolená, keď sa posiela push notifikácia.
406	Cloud service vyčerpala limit pre notifikácie. Server by zredukovať množstvo požiadavok na notifikácie.
410	Kanál expiroval. Aplikácia by mala požiadať o nový URI.
413	Notifikácia prekročila limit 5000 bajtov.
500	Interná serverová chyba.
503	Server je nedostupný.

Tabuľka 8: WNS response

Apple Push Notification Service:

Status kód	Popis
0	Žiadna chyba nenastala.
1	Chyba pri spracovaní požiadavku.
2	Chýbajúci device token.
3	Chýbajúci obsah.
4	Chýbajúci payload

5	Nesprávna veľkosť tokenu.
6	Nesprávna veľkosť obsahu.
7	Nesprávna veľkosť payload.
8	Nesprávny token.
255	Neznáma chyba.

Tabuľka 9: APNS response

Odpovede zo serverov nie sú totožné a vyžadujú si preto odlišnú implementáciu pre spracovanie chybových správ. Serveri pre Windows Phone a Windows 8 sú veľmi podobné. Windows Notification Services je novší a obsahuje väčší počet možných odpovedí. Notifikačný server pre Android obsahuje najmenej rôznych odpovedí, ale správa 200 indikujúca úspešné spracovanie môže obsahovať chybové správy pri preskúmaní detailov správy. APNS je podobný ako Windows servery.

5.2 Azure mobile services

Jedna zo špeciálnych služieb , ktoré ponúka Windows Azure je služba Mobile Services. Je dostupná s príchodom nového portálu v beta verzii. Služba ma po dokončení umožňovať posielanie notifikácii bez ohľadu na platformu. Jej hlavným cieľom je uľahčiť vývoj mobilných aplikácii od zložitého implementovania notifikácii pre rôzne platformy a to už naimplementovanými funkciami, ktoré je potrebné len nakonfigurovať. Mobile Services ponúkajú nasledujúcu funkcionalitu:

1. Dátové služby – možné využitie napríklad pre ukladanie zdieľaných dát medzi zariadeniami, alebo ich uchovávať aby sa zachovali pri ztráte zariadenia.
2. Autentizačné služby – v prípade, že nechceme využívať svoju autentizáciu, môžeme využiť overenie užívateľa voči Facebooku, LiveID, Google, Twitter,... Táto varianta je pomerne zložitá na implementáciu, preto je ich použitie pripravené v Mobile Services
3. Notifikačné služby – jednoduché používanie notifikácii pre mobilné platformy

Dáta sa ukladajú do SQL databáze a pripravené funkcie vykonáva serverový JavaScript.

Táto služba je pri písaní tejto práce v beta verzii a podporuje zatiaľ Windows Phone 7, Windows Phone 8 a iOS platformu. Poplatky za využívanie tejto služby zatiaľ nie sú známe, ale je možné ich vyskúšanie v rámci Azure účtu.

Z pohľadu ďalších komerčných aplikácii existuje na trhu viacero spoločností ponúkajúcich ich software pre posielanie notifikácii na rôzne platformy ako napríklad Push Woosh, Push IO a iné. Podporujú rôzne platformy a ich cenové politiky sú taktiež rôzne. Niektoré ponúkajú svoje služby zdarma s obmedzeniami ako napríklad maximálny počet zariadení, alebo obmedzený počet notifikácii za určité časové obdobie.

6 Bezpečnosť architektúry pre jednotlivé platformy

Architektúra push notifikácii je rôzna pre jednotlivé platformy. Z veľkej časti ide o zabezpečenú HTTPS komunikáciu medzi zariadením a príslušným notifikačným serverom alebo samotným koncovým zariadením a notifikačným serverom. Každá platforma ale pristupuje k zabezpečeniu pomocou HTTPS odlišne a vyžaduje rozdielnu implementáciu a autentifikáciu.

Komunikácia zariadenia s webovým serverom na ktorom beží určitá logika našej aplikácie je nezabezpečená respektíve je na tvorcoch aplikácie ako zabezpečia túto časť architektúry. A táto komunikácia je nutná pre každú platformu, ako vyplýva z architektúry.

HTTPS

Hypertext Transfer Protocol Secure je nadstavba sieťového protokolu http, ktorá umožňuje zabezpečiť spojenie medzi webovým prehliadačom a webovým serverom pred odposluchávaním, podhadzovaním dát a umožňuje taktiež overiť identitu protistrany. HTTPS používa protokol HTTP, pričom prenášané dáta sú šifrované pomocou SSL alebo TLS a štandardný port na strane serveru je 443.

SSL

Secure Sockets Layer, je vrstva vložená medzi vrstvu transportnú (napríklad TCP/IP) a aplikačnú (napríklad HTTP), ktorá poskytuje zabezpečenie komunikácie šifrovaním a autentizáciou komunikujúcich strán.

Protokol SSL sa najčastejšie využíva pre bezpečnú komunikáciu s internetovými servermi pomocou HTTPS. Po vytvorení SSL spojení je komunikácia medzi serverom a klientom šifrovaná a teda zabezpečená.

6.1 iOS

Podobne ako aj Android zariadenie, aj iOS zariadenie potrebuje podobný kľúč od notifikačného serveru APNS. Táto komunikácia je zabezpečená SSL a certifikátom, ktorý je vytvorený pre danú aplikáciu pre notifikácie na Apple portáli. Ten istý certifikát je potrebný aj na komunikáciu aplikačného serveru, ktorý odosiela notifikácie na APNS.

Podobne ako aj u Androidu, jediným miestom ktoré nie je zabezpečené APNS serverom je komunikácia medzi zariadením a naším serverom, ktorý generuje notifikácie. Preto ak by sme chceli poslať token, mohol by ho niekto odchytiť na sieti. Na to by ale potreboval ešte samotný certifikát, bez neho by mu bol token k ničomu. Tuto komunikáciu si tiež môžeme zabezpečiť pomocou HTTPS s overeným certifikátom.

Nasledujúca ukážka kódu v Objective C zobrazuje HTTPS request. Certifikát, ktorý používa server je certifikát vydaný od známej Certifikačnej Autority, takže netreba dodatočnú implementáciu.

```
NSURL *url = "https://...";
NSURLRequest *urlRequest = [NSURLRequest requestWithURL:url
cachePolicy:NSURLRequestReturnCacheDataElseLoad
timeoutInterval:30];
NSData *urlData;
NSURLResponse *response;
NSError *error;
urlData = [NSURLConnection sendSynchronousRequest:urlRequest
returningResponse:&amp;response
error:&amp;error];
```

Zdrojový kód 28: iOS HTTPS

6.2 Android

Toto zariadenie potrebuje získať ID od notifikačného GCM serveru, táto komunikácia je zabezpečená pomocou SSL. Aj samotný notifikačný server ktorý posiela notifikáciu na GCM server, komunikuje s ním taktiež v zabezpečenom SSL režime.

Jediným miestom ktoré nie je zabezpečené google serverom je komunikácia medzi zariadením a naším serverom, ktorý generuje notifikácie. Preto ak by sme chceli poslať Device ID, mohol by ho niekto odchytiť na sieti. V zásade by potreboval ešte API key a dúfať, že aplikácia má povolené na posielanie notifikácii všetky IP adresy. Čo by ale mohol odchytiť, je prípadná komunikácia medzi naším zariadením a naším serverom, ak by sme mu posielali aj nejaké citlivé dáta. Tuto komunikáciu si tiež môžeme zabezpečiť pomocou HTTPS s overeným certifikátom.

Nasledujúca ukážka kódu v Jave zobrazuje HTTPS request. Certifikát, ktorý používa server je certifikát vydaný od známej Certifikačnej Autority.

```
URL url = new URL("https://...");
URLConnection urlConnection = url.openConnection();
InputStream in = urlConnection.getInputStream();
copyInputStreamToOutputStream(in, System.out);
```

Zdrojový kód 29: Android HTTPS

6.3 Windows 8

Získanie kľúču z aplikácie voči WNS serveru je zabezpečené SSL certifikátom, ktorý musí mať aplikácia zaregistrovaný na Windows Marketplace pre danú aplikáciu. Serverová aplikácia je nezávislá na použitej platforme, pretože WNS používa na komunikáciu štandardne protokoly REST alebo JSON, rovnako ako aj Android a iOS. Server sa musí najskôr autentifikovať voči WNS. Potrebuje nato SID a SecretKey. Následne tieto dáta pošle cez HTTPS request na WNS server. Po získaní autentifikačného kľúču stačí už len poslať samotnú notifikáciu so všetkými potrebnými údajmi na WNS server. Rovnako ako Android a iOS, aj vo Windows 8 aplikáciach je jediné nezabezpečené miesto komunikácia zariadenia s naším serverom pre vytváranie notifikácii. Windows RT ma sprístupnené takmer celé API desktopového Windows. Môžeme ho teda zabezpečiť rovnako ako desktopové aplikácie alebo aj pomocou SSL a ukážka volania takejto servisy je nasledujúce zo C# kódu:

```
WebRequest webRequest = WebRequest.Create("https://...");
WebResponse webResp = webRequest.GetResponse();
```

Zdrojový kód 30: Windows 8 HTTPS

6.4 Windows Phone

Pre získanie URI z MPNS nepotrebuje aplikácia certifikát, komunikácia prebieha cez nezabezpečené HTTP a obsahuje svoje denné limity pre notifikácie. Táto komunikácia sa dá zabezpečiť zaregistrovaním certifikátu podobne ako pre Windows RT.
Získané URI zašle aplikácia na server. Samotný server sa nemusí autentifikovať voči MPNS, stačí mu URI, preto sú Windows Phone 7 a 8 notifikácie asi najmenej zabezpečené. Stačí odchytiť URI, ktoré prislúchá danému zariadeniu a možeme posielať push notifikacie z akého koľvek serveru. Samozrejme to neplatí pre aplikácie používajúce overený a zaregistrovaný certifikát. V takom prípade je zabezpečenie podobné iOS notifikáciam, kedy je potrebné použiť tento certifikát voči MPNS.

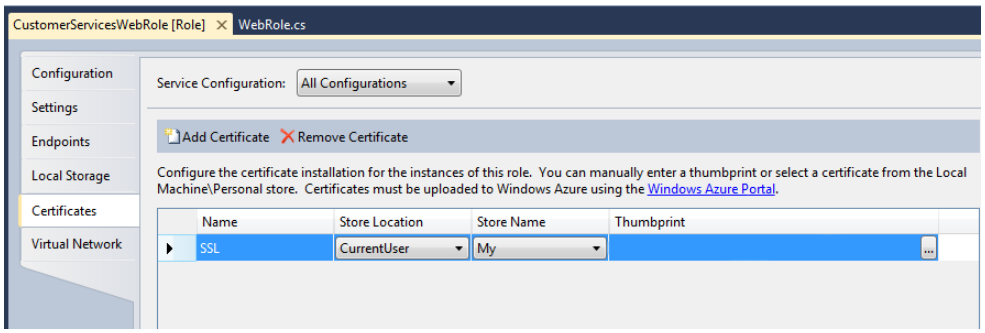
Tak ako Android, iOS a Windows 8 aj vo Windows Phone aplikáciach je jediné nezabezpečené miesto komunikácia zariadenia s naším serverom pre vytváranie notifikácii. Pri nezabezpečených notifikáciach certifikátom je ale obzvlášť nebezpečné, keďže sa nemusí server vôbec autentifikovať voči MPNS. Windows Phone a najmä Windows Phone 8 ma sprístupnené takmer celé API desktopového Windows. Môžeme ho teda zabezpečiť rovnako ako desktopové aplikácie alebo aj pomocou SSL a ukážka volania takejto servisy je identická ako pre Windows RT.

6.5 Windows Azure a zabezpečenie pomocou HTTPS

Všetky platformy zabezpečujú svoje notifikácie nanajvýš pomocou SSL cez protokol HTTPS, preto by sa mal aj samotný aplikačný Azure server, ktorý vytvára notifikácie zabezpečiť pomocou HTTPS.

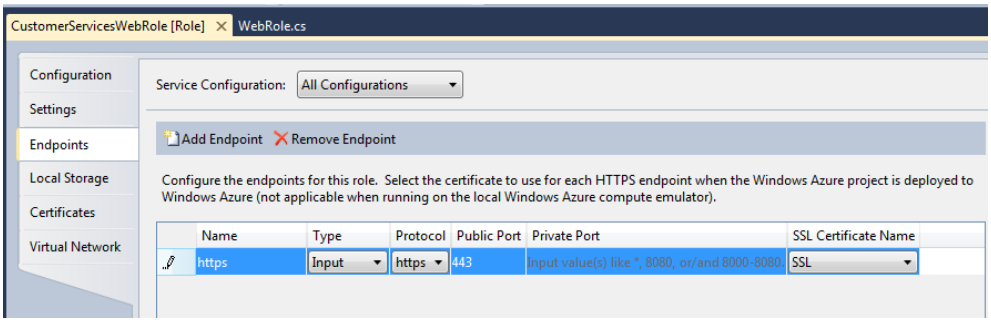
Na podporu takto šifrovanej komunikácie je potrebný SSL certifikát od Certifikačnej Autority pre moju doménu aplikácie. Vydanie tohto certifikátu stojí približne 100$, preto som použil certifikát, ktorý si sami podpíšeme. Tento certifikát nefunguje ako overený certifikát, ale nasadiť zakúpený a overený certifikát namiesto môjho už nie je problém.

Certifikát je potrebné nahrať na Azure portál do notifikačnej aplikácie v sekcii nahrávania certifikátov. Následne jeho inštalácia na počítač na ktorom som vyvíjal Azure aplikáciu a to dvojklikom na certifikát a inštalovať. Inštaláciu overíme v programe certmgr.msc. Nájdeme ho pod priečinkom Personal, teda v našom certifikačnom liste, nie pod certifikátmi počítača. Visual Studio ale prehľadáva len certifikáty počítača, keď ho budeme chcieť pridať do Azure konfigurácie nášho riešenia. Visual Studio ale potrebuje len thumbprint, ktorý je vo vlastnostiach tohto certifikátu, stačí len zmazať medzery a zmeniť malé písmena na veľké. Konfigurácia HTTPS sa nastavuje v nastaveniach projektu, pod záložkou certificates, kde sa vloží pripravený thumbprint.



Obrázok 29: Windows Azure SSL

Po tomto nastavení môžeme použiť certifikát pod záložkou endpoints, kde je prístupný pod certifikátmi.



Obrázok 30: Windows Azure Endpoint

Naša aplikácia je WCF, preto musíme upraviť aj web.config pre podporu SSL. Nemôžeme nainštalovať certifikát na každé zariadenie, ktoré bude komunikovať s WCF servisov, preto sme použili Transport security bez prihlasovacích údajov.

```
<basicHttpBinding>
      <binding name="secureHttpBinding">
            <security mode="Transport">
                  <transport clientCredentialType="None" />
            </security>
      </binding>
</basicHttpBinding>
```

Zdrojový kód 31: Bezpečnost HttpBinding

Taktiež musíme zmeniť pôvodný httpGetEnabled na httpsGetEnabled.

```
<serviceBehaviors>
      <behavior>
            <serviceMetadata httpsGetEnabled="true"/>
            <serviceDebug includeExceptionDetailInFaults="false"/>
      </behavior>
</serviceBehaviors>
```

Zdrojový kód 32: Bezpečnosť serviceBehavior

Po týchto nastaveniach a s overeným certifikátom, by sme mohli použiť Azure WCF servisu v zabezpečenom HTTPS režime. Tým by bola celá architektúra zabezpečená pomocou HTTPS a SSL.

7 Výkonnostné testy notifikácii

Architektúra notifikácii je podobná pre všetky platformy, preto aj ich výkonnosť by mala byť podobná. Rozdiely ale môže spôsobiť reakcia koncového zariadenia na prijatú notifikáciu. Preto v tejto sekcii budem overovať ich výkonnosť. Testovaný je Android, Windows Phone 7, Windows Phone 8. Windows 8 a iOS som netestoval kvôli potrebe zakúpiť certifikát. Testy su opakované trikrát a výsledný čas je priemerom nameraných časov. Čas je meraný softwarovými stopkami, ktoré sa spustia s odosielaním na notifikačný server. Zariadenia sú výkonnostne podobné. Odoslanie 100 správ na jedno zariadenie na každú platformu:

	Windows Phone 7	Windows Phone 8	Android
Toast	320 sekúnd	365 sekúnd	37 sekúnd
Normal View	-	-	35 sekúnd
Tile	22 sekúnd	21 sekúnd	-
Raw	15 sekúnd	15 sekúnd	-

Tabuľka 10: Výkonnostné testy 1

Z výsledkov notifikácii sú Windows Phone 7 aj Windows Phone 8 výkonnostne takmer identické. Toast notifikácie sú ale veľmi pomalé v porovnaní s Androidom. Toast notifikácie vo Windows Phone sa viac podobajú na Normal View notifikácie pre Android, ale ten má oba druhy notifikácii časovo takmer zhodné. Naopak tile a raw notifikácie pre Windows Phone su výkonnostne pred Androidom. Ďalším testom bolo odoslanie 100 správ na dve zariadenia súčastne pre každú platformu, výsledný čas je priemerným časom z dvoch zariadení:

	Windows Phone 7	**Windows Phone 8**	**Android**
Toast	342 sekúnd	373 sekúnd	39 sekúnd
Normal View	-	-	34 sekúnd
Tile	23 sekúnd	24 sekúnd	-
Raw	15 sekúnd	17 sekúnd	-

Tabuľka 11: Výkonnostné testy 2

Výsledné časy sú takmer identické s predchádzajúcim testom. Tieto notifikačné servery sú pripravené pre vysoký počet požiadavkou, preto sú časy takmer identické s predchádzajúcim testom. Test by sa mohol zopakovať voči desiatkam zariadení, kde by sa časové rozdiely možno prejavili.

8 Záver

Som rád, že som si mohol vyskúšať vývoj pre všetky spomínané platformy, aj napriek problémom s potrebnými certifikátmi. Porovnanie typov notifikácii jednotlivých platforiem zobrazuje rozdiely v týchto platformách, podobne ako aj porovnanie architektúry posielania notifikácii. Výkonnostné testy zobrazujú možné úpravy notifikácii pre čo najvyšší výkon.

Bezpečnosť notifikácii je jedna z ich najväčších nevýhod. V tejto diplomovej práci som sa zaoberal aj touto problematikou a porovnával architektúru notifikácii aj z bezpečnostného hľadiska. V práci som popisoval možné zabezpečenie nezabezpečených miest architektúry notifikácii.

Push notifikácie sa neustále menia a zjednodušuje sa ich vývoj. Windows Azure ponúka množstvo funkcionality, ktorá urýchľuje vývoj aplikácii. Zaujmavá možnosť by bola vyskúšať Windows Azure Mobile Servises po ich dokončení.

9 Literatúra

[1] Developer Apple, *Document Revision History*, 2013, <http://developer.apple.com/library/ios/#releasenotes/General/WhatsNewIniOS/RevisionHistory.html#//apple_ref/doc/uid/TP40008246-CH99-SW1>.

[2] MSDN, *Push notification overview*, 2012, <http://msdn.microsoft.com/en-us/library/windows/apps/hh913756.aspx>.

[3] Developer Android, *Getting Started*, 2013, <http://developer.android.com/training/index.html>.

[4] Developer Android, *Notifications*, 2013, <http://developer.android.com/guide/topics/ui/notifiers/notifications.html>.

[5] Windows Phone, *Windows Phone*, 2012, < http://www.windowsphone.com/en-us/features >.

[6] Tech Terms, *iOS*, 2012, < http://www.techterms.com/definition/ios >.

[7] MSDN, *Windows Azure*, 2012, <http://www.microsoft.com/slovakia/azure/windowsazure/>.

[8] Neil Mackenzie, *Microsoft Windows Azure Development Cookbook*, 2011

[9] Google, *GCM Architectural Overview*, 2012, <http://developer.android.com/google/gcm/gcm.html>.

[10] Codeplex, *Project Hosting for Open Source Software*, 2006, < http ://www.codeplex.com>.